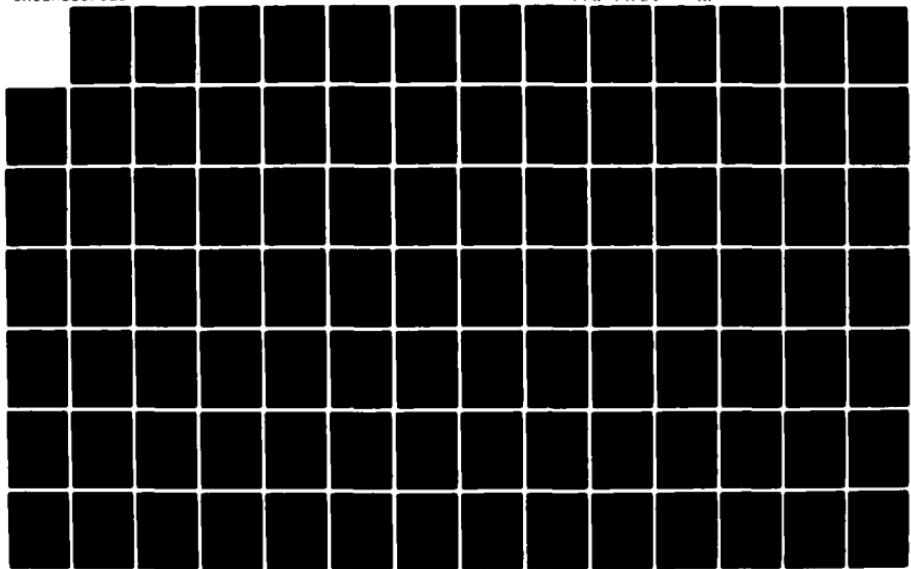


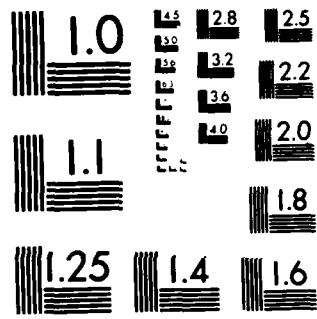
AD-A126 980 A MICROCOMPUTER-BASED SAMPLING DIGITAL VOLTmeter(U)  
NAVAL RESEARCH LAB WASHINGTON DC R E SCOTT ET AL.  
31 MAR 83 NRL-MR-4872

1/2

UNCLASSIFIED

F/G 14/2+ NI





MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

*(Handwritten mark)*  
NRL Memorandum Report 4872

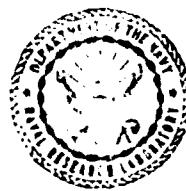
ADA126980

## A Microcomputer-Based Sampling Digital Voltmeter

RICHARD E. SCOTT, JR. AND JAMES D. GEORGE

*Acoustical Systems Branch  
Underwater Sound Reference Detachment  
Naval Research Laboratory  
P.O. Box 8337  
Orlando, Florida 32856*

March 31, 1983



NAVAL RESEARCH LABORATORY  
Washington, D.C.

Approved for public release; distribution unlimited.

83 04 19 069

DTIC FILE COPY

## UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM												
1. REPORT NUMBER NRL Memorandum Report 4872	2. GOVT ACCESSION NO. AD-A126 880	3. RECIPIENT'S CATALOG NUMBER												
4. TITLE (and Subtitle) A Microcomputer-Based Sampling Digital Voltmeter		5. TYPE OF REPORT & PERIOD COVERED Interim report on a continuing problem												
		6. PERFORMING ORG. REPORT NUMBER												
7. AUTHOR(s) Richard E. Scott, Jr. and James D. George		8. CONTRACT OR GRANT NUMBER(S)												
9. PERFORMING ORGANIZATION NAME AND ADDRESS Underwater Sound Reference Detachment Naval Research Laboratory P.O. Box 8337, Orlando, FL 32856		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element: 62711N Project NRL Work Unit: (59)-0593												
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command (NAVSEA 63R12) Washington, DC 20362		12. REPORT DATE 31 March 1983												
14. MONITORING AGENCY NAME & ADDRESS/If different from Controlling Office)		13. NUMBER OF PAGES 109												
		15. SECURITY CLASS. (of this report) UNCLASSIFIED												
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A												
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.														
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, If different from Report)														
18. SUPPLEMENTARY NOTES This work was performed in the Acoustic Metrology Program sponsored by the Naval Sea Systems Command (SEA63R12).														
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">Vector voltmeter</td> <td style="width: 33%;">Distortion measurement</td> <td style="width: 33%;">Kaiser window</td> </tr> <tr> <td>True rms</td> <td>A/D converter</td> <td>Microcomputer</td> </tr> <tr> <td>Phase measurement</td> <td>SFDFT</td> <td>PDP-11</td> </tr> <tr> <td></td> <td></td> <td>IEEE-488 bus</td> </tr> </table>			Vector voltmeter	Distortion measurement	Kaiser window	True rms	A/D converter	Microcomputer	Phase measurement	SFDFT	PDP-11			IEEE-488 bus
Vector voltmeter	Distortion measurement	Kaiser window												
True rms	A/D converter	Microcomputer												
Phase measurement	SFDFT	PDP-11												
		IEEE-488 bus												
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes a sampling digital voltmeter developed at the Underwater Sound Reference Detachment of the Naval Research Laboratory. It consists of a PDP-11/23 microcomputer with up to three channels of analog-to-digital converters capable of making simultaneous measurements on pulsed or continuous sinusoids at frequencies up to 100 kHz. The voltmeter is controlled by ASCII commands on the IEEE-488 bus. Statistical parameters (mean, peak, standard deviation, and second joint moment) are computed using numerical integration. Sinusoidal parameters (amplitude, phase, and harmonic (continued)														

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE  
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

**UNCLASSIFIED**

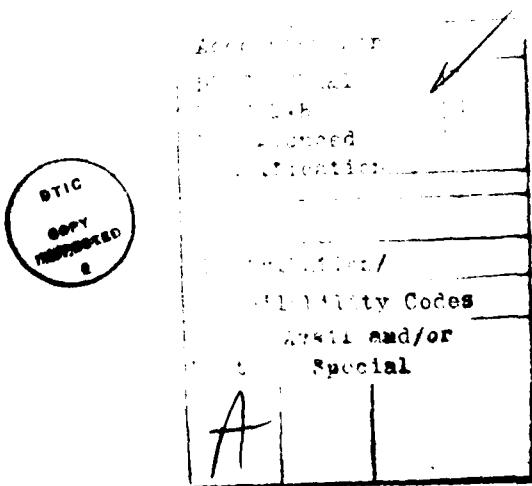
**SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)**

(Item 20 continued)

distortion) are computed using single-frequency discrete Fourier transforms (SFDFT). The digital filtering resulting from the use of the SFDFT can be further enhanced by optional data sequence weighting (multiplying the sequence by a given windowing sequency). Computation times may be reduced by optional subsequence averaging. Complete assembly language listings for the voltmeter program are included and described.

## CONTENTS

INTRODUCTION.....	1
APPLICATIONS.....	1
THEORY OF OPERATION.....	2
COMPUTATION PROCEDURES.....	4
Sampling Criteria.....	4
Sinusoidal Parameters.....	6
Digital Filtering.....	7
Statistical Parameters.....	10
SOFTWARE FUNCTIONS.....	11
SUMMARY .....	13
ACKNOWLEDGMENTS.....	14
REFERENCES.....	14
APPENDIX A - Voltmeter Hardware.....	15
APPENDIX B - Voltmeter Commands.....	17
APPENDIX C - Voltmeter Status Messages.....	21
APPENDIX D - Voltmeter Control Subroutines.....	23
APPENDIX E - Voltmeter Program Listings.....	31
APPENDIX F - Voltmeter Memory Map.....	109



## A MICROCOMPUTER-BASED SAMPLING DIGITAL VOLTmeter

### INTRODUCTION

Acoustical measurement systems at the Underwater Sound Reference Detachment (USRD) of the Naval Research Laboratory need a specialized multiple-channel voltmeter with the capability to measure voltage, relative phase between channels, and harmonic distortion for pulsed sinusoids at frequencies between 0.1 Hz and 100.0 kHz. To answer this need, a sampling digital voltmeter was developed around an LSI-11/23 microcomputer, using three analog-to-digital (A/D) converters and an IEEE-488 interface to serve as the communications link with any appropriate host computer. Using this voltmeter, a "simple" computer-controlled measurement system can then be configured with the following auxiliary equipment as shown in Fig. 1: synthesizers to generate a sinusoidal test signal and a coherent sampling signal, and programmable amplifiers or attenuators as needed. (The voltmeter has no intrinsic autoranging capability.)

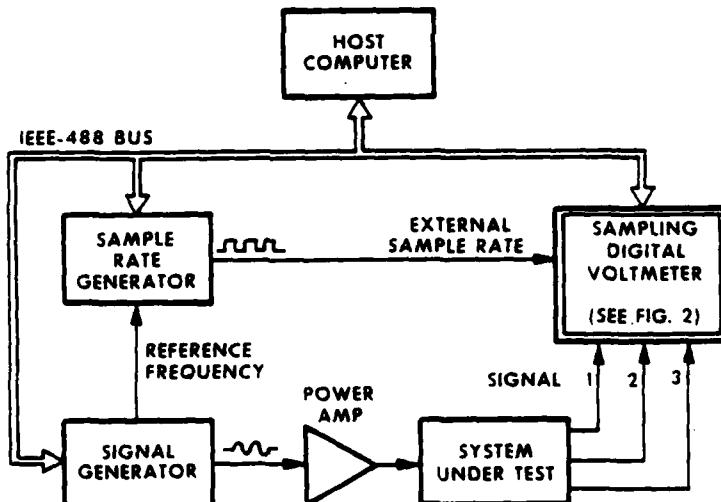


Fig. 1 - Typical measurement system using the microcomputer-based sampling voltmeter.

### APPLICATIONS

The two immediate applications for such a voltmeter were for the USRD's Anechoic Tank Facility (ATF) and Low-Frequency Facility (LFF) measurement systems.

The ATF high-power measurement system requires a three-channel voltmeter to measure simultaneous projector drive voltage, drive current, and receiver output (hydrophone voltage). Measurements of relative phase between voltage and current and the harmonic distortion in all three channels are necessary. The practical frequency range is 2 to 100 kHz; the mode of operation is usually pulsed.

The LFF system requires a two-channel voltmeter to measure the simultaneous voltage outputs of two hydrophones, generally a probe standard and an unknown hydrophone. Relative acoustic phase is required, and harmonic distortion is a useful parameter to monitor. The practical frequency range is 0.1 Hz to 4 kHz; the mode of operation is continuous wave (cw).

The voltmeter was, therefore, conceived to be a device independent of (and transportable between) specific measurement systems at the USRD, with the possible exception being that the A/D converters might vary with application (e.g., a need to operate at higher sampling rates) and are otherwise functionally identical plug-in modules. To get the widest possible range of potential host computers, communication through the IEEE-488 bus was chosen. Another requirement was that all external hardware, including the sample-rate generator, be controlled by the host computer for maximum flexibility. Therefore, all external functions, including automatic ranging of the voltmeter, are controlled by the host.

Since initiating and implementing this signal processing sampling voltmeter, two other similar instruments have been described or released: the low-frequency sampling voltmeter by the National Bureau of Standards [1] and the sampling network analyzer by the Dranetz Company [2]. The three instruments share the sampling ideas but differ in the features and algorithms implemented.

#### THEORY OF OPERATION

The voltmeter consists of a microcomputer containing up to three A/D converters with direct memory access (DMA) capability (Fig. 2). This configuration allows the computer to digitize up to three input waveforms simultaneously, sampling them when triggered by a common external sample-rate generator. Where simultaneous measurement of all channels is not desired, separate or delayed sampling signals can be given to individual A/D converters. The present voltmeter is designed to digitize three waveforms of up to 1024 samples each; these buffers could be expanded to 3900 samples (see Appendix F).

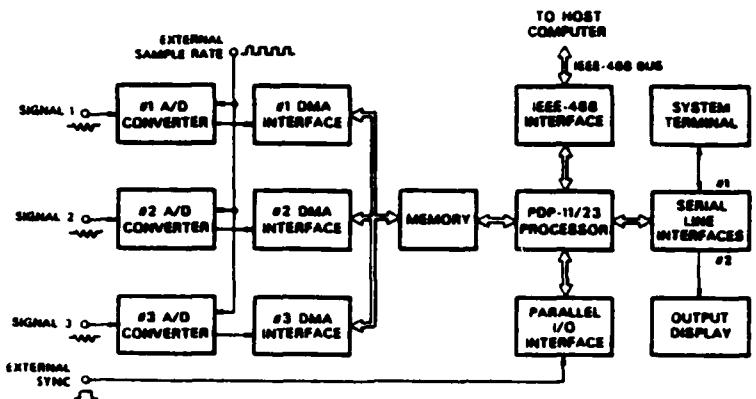


Fig. 2 - Block diagram of the microprocessor-based sampling voltmeter.

The DMA interface loads 12-bit digitized values directly into memory at rates up to 100 kHz per channel. The individual sampled waveforms can be transmitted to the host for processing; alternately, the slave can compute sinusoidal and statistical parameters from these waveforms and return them to the host. Parameters are computed for the fundamental frequency. (Throughout this report "fundamental" frequency refers to the sinusoidal test frequency being measured, no matter how many cycles of this frequency are included in a measured sequence.)

The sinusoidal quantities amplitude, phase, and harmonic distortion are computed for each channel using single frequency discrete Fourier transforms (SFDFT). The statistical quantities average, standard deviation, and the positive and negative extrema are obtained for each channel to characterize noisy or nonsinusoidal signals. In voltmeter terms, these statistical parameters are the dc, true ac rms, and positive and negative peak voltages. To measure the coherence or correlation between channels, the second order joint moment statistic,  $E(xy)$ , can be computed. For the case of the two signals,  $x$  and  $y$ ; proportional to current and voltage, the joint moment is proportional to power. The covariance and cross correlation can be readily obtained from these parameters.

## COMPUTATION PROCEDURES

### Sampling Criteria

In the usual case of interest (pulsed sinusoids), sampling errors are minimized through coherent sampling, which requires an integer number of both test frequency cycles and sampling periods:

$$\frac{m}{f_T} = \frac{n}{f_S} , \quad (1)$$

where  $f_T$  is test frequency

$f_S$  is sampling frequency

$n$  is samples per sequence (integer)

$m$  is cycles per sequence (integer).

If the sampling frequency,

$$f_S = \frac{n}{m} f_T , \quad (2)$$

is greater than twice the test frequency, the Nyquist theorem is satisfied, and satisfactory results will be obtained. If the necessary sampling rate is too fast for the A/D converter to operate properly, undersampling can be used; by choosing fewer than two samples per cycle while retaining an integer number of samples and cycles per sequence, an effective higher sampling rate is attained. A comparison of the oversampled waveform shown in Fig. 3 with the undersampled waveform in Fig. 4 clearly demonstrates how undersampling can synthesize a waveform (Fig. 5) effectively sampled at a higher rate, although with fewer cycles overall. Undersampling will only work properly on periodic waveforms, however. Computed data in Table 1 demonstrates that peak voltage and harmonic distortion may be affected slightly by undersampling.

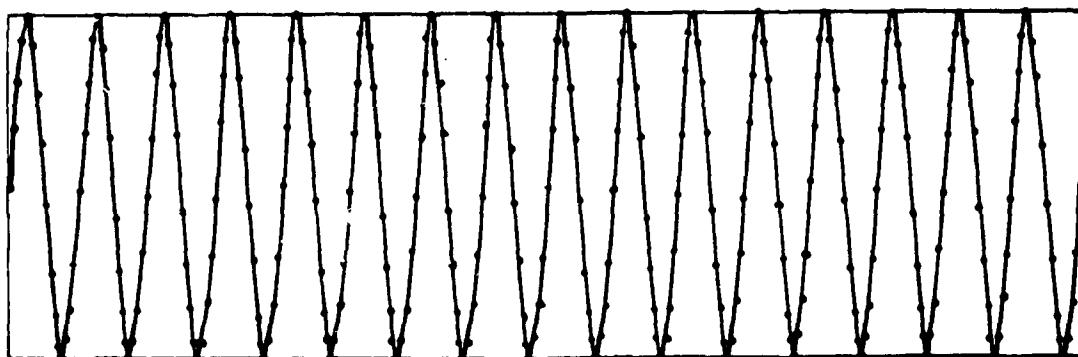


Fig. 3 - Steady-state sinusoidal waveform sampled for 16 cycles at 17 samples per cycle: 272 samples (oversampling).

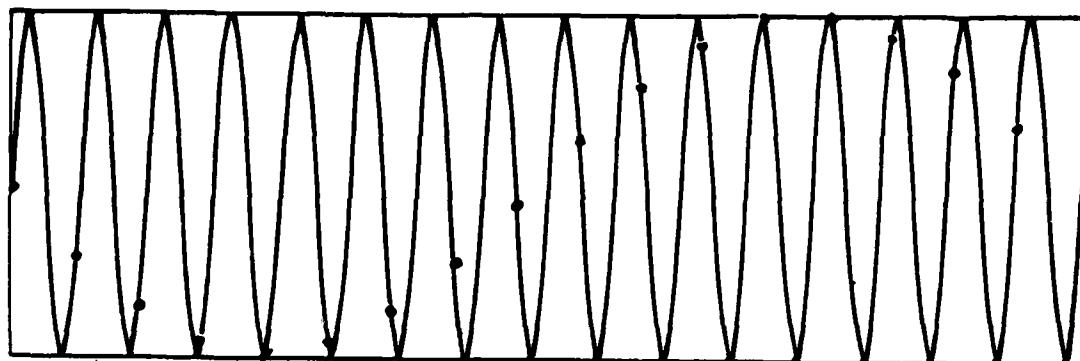


Fig. 4 - Steady-state sinusoidal waveform sampled for 16 cycles at 1.0625 samples per cycle: 17 samples (undersampling).

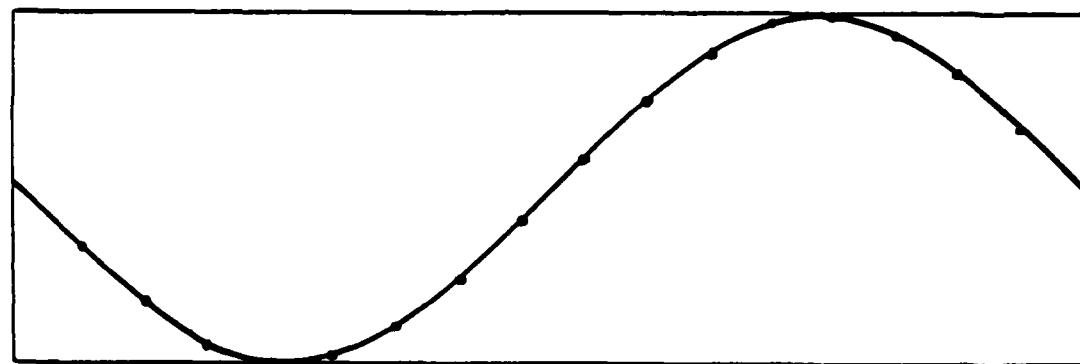


Fig. 5 - Steady-state sinusoidal waveform synthesized from the 17 samples shown in Fig. 5.

Table 1 - Comparison of oversampled and undersampled measurements.

COMPUTED PARAMETER	OVER SAMPLING	UNDER SAMPLING
RMS VOLTAGE (STD DEV)	1.047	1.047
PEAK VOLTAGE	1.485	1.492
DC VOLTAGE (MEAN)	-0.013	-0.013
TOTAL POWER	1.102	1.102
RMS VOLTAGE (DFT)	1.047	1.047
RELATIVE PHASE	0	0
HARMONIC DISTORTION	0.100	0.095

Normal: 2 cycles at 26 samples per cycle (52 samples total).

Undersampled: 25 cycles at 1.04 samples per cycle (52 samples total).

### Sinusoidal Parameters

For sinusoidal testing, the signals are of the form

$$y(t) = \sqrt{2}A_m \cos(2\pi f_T t + \phi_m) + \sum_{k=1}^l \sqrt{2}A_{km} \cos(2\pi kf_T t + \phi_{km}) + n(t) , \quad (3)$$

where the terms are, respectively, the fundamental, the harmonics, and the additive noise.

Least squares estimates of the rms amplitude,  $A_m$ , and phase,  $\phi_m$ , of the fundamental are obtained from

$$\hat{A}_m = \sqrt{2} [\operatorname{Re}^2(Y_m) + \operatorname{Im}^2(Y_m)]^{1/2} , \quad (4)$$

and

$$\hat{\phi}_m = \tan^{-1} \left[ \operatorname{Im}(Y_m) / \operatorname{Re}(Y_m) \right] , \quad (5)$$

where

$$Y_m = \frac{1}{n} \sum_{i=0}^{n-1} y_i \exp(-j2\pi im/n) , \quad (6)$$

which is the  $m$ th component of the DFT of the  $n$ -point sequence  $y_i$  containing  $m$  cycles of the fundamental; that is, the single frequency discrete Fourier transform (SF DFT). With coherent sampling and in the absence of noise,  $A_m$  and  $\phi_m$  are exact.

Harmonic distortion,  $D$ , is defined in terms of the ratio of power in the harmonics to power in the fundamental.

$$D = \left[ \sum_{k=2}^l A_{km}^2 / A_m^2 \right]^{1/2} . \quad (7)$$

Since the sequences are relatively short and few lines are required, Goertzel's algorithm [3,4] is an efficient implementation of the SFDFT.

#### Digital Filtering

The evaluation of the DFT at a single frequency, as in Eq. (6) can be interpreted as a digital filter [5] with a corresponding enhanced signal-to-noise ratio. However, the filter quality, the skirts and side lobes, can be significantly improved by weighting or windowing the data sequence and then doing the SFDFT. A particularly effective choice of data weighting sequence is the Kaiser  $I_0$ -sinh window [6]. An example of a weighted data sequence is shown in Fig. 6, and the corresponding digital filter responses are shown in Fig. 7.

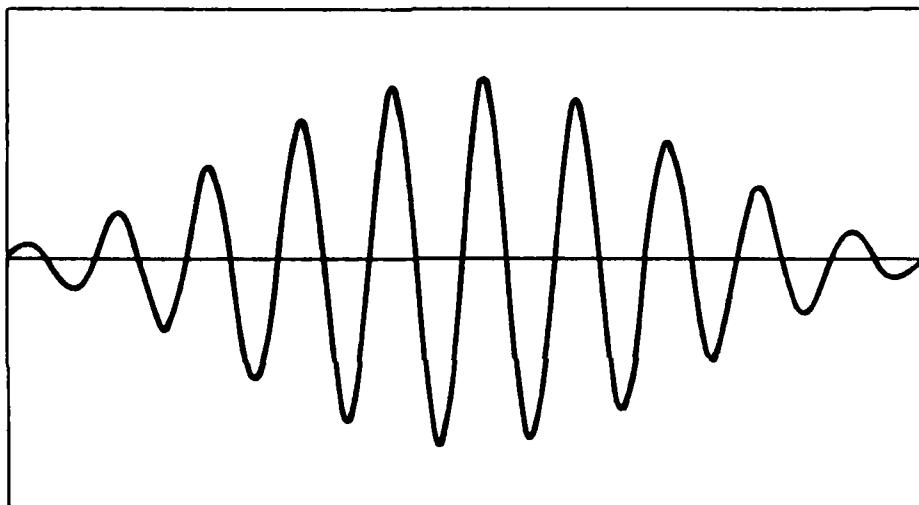


Fig. 6 - Sinusoidal waveform of 10 cycles (product with Kaiser window with 40 dB of spectral side lobe attenuation).

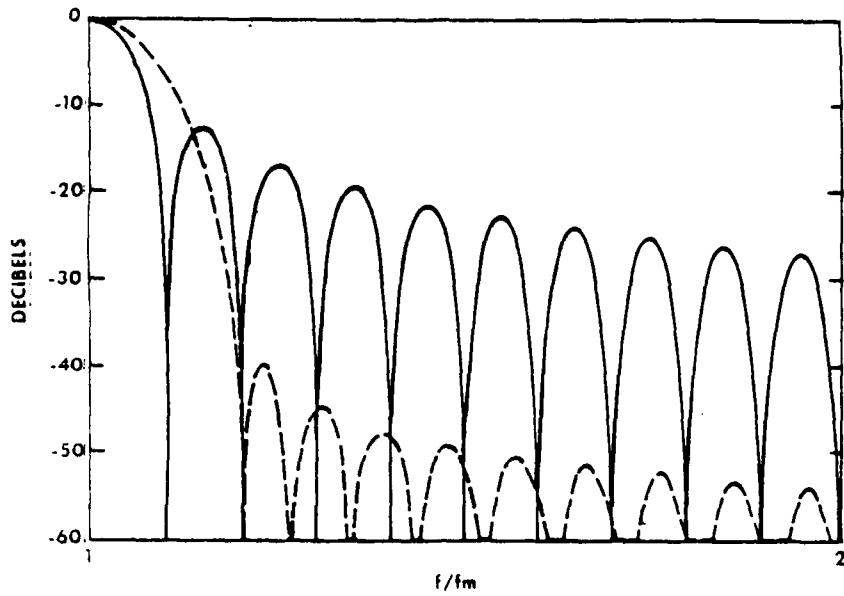


Fig. 7 - Comparison of the filter effect of an unweighted SFDFT (solid line) with that of a SFDFT with the Kaiser window with 40 dB of side lobe attenuation (dashed line). The frequency axis is normalized to the center frequency  $f_m$  which corresponds to the  $m$ th spectral line. The data sequence includes 10 cycles.

The filters are centered on the  $m$ th spectral line or test frequency,  $f_T$ . Following convention, the filter bandwidth is defined as the main lobe width or the distance between the minima adjacent to the  $m$ th line. For the unweighted SFDFT, the fractional bandwidth or main lobe width is

$$\left( \frac{\Delta f}{f_m} \right)_{\text{Rectangular}} = \frac{2}{m} , \quad (8)$$

where  $m$  is the number of cycles of  $f_T$  in the data sequence.

The reduced side lobes of the Kaiser filter are achieved at the expense of an increased main lobe width

$$\left( \frac{\Delta f}{f_m} \right)_{\text{Kaiser}} = \left( \frac{2}{m} \right) \frac{6(R + 12)}{155} , \quad (9)$$

where  $R$  is the attenuation in dB of the first side lobe. Figure 8 illustrates the increase in filter main lobe width as the side lobes are reduced.

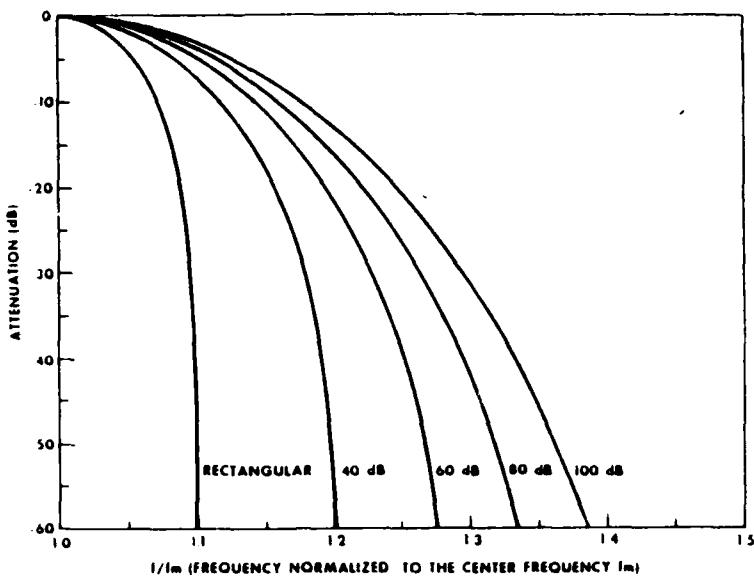


Fig. 8 - Comparison of filter main lobes for unweighted SFDFT and Kaiser-weighted SFDFT with 40 to 100 dB side lobes. The side lobes are not plotted for clarity. The data sequence includes 10 cycles ( $m = 10$ ).

The data window or weighting sequence,  $w_i$ , is computed in the host and transferred to the voltmeter where the window is applied in estimating amplitude and phase using the SFDFT of the weighted sequence

$$Y_m = \frac{1}{n} \sum_{i=0}^{n-1} w_i y_i \exp(-j2\pi i m/n) . \quad (10)$$

The amplitude estimate,  $A_m$ , is corrected by the window scale factor

$$\bar{w} = \frac{1}{n} \sum_{i=0}^{n-1} w_i . \quad (11)$$

Table 2 illustrates that the Kaiser window does not alter or bias the amplitude and phase estimates at the test frequency. However, the harmonic distortion, particularly for low distortion, is biased by the window; larger sidelobe attenuations reduce this bias by reducing the amplitude of the harmonics created by the window.

Table 2 - Amplitude, phase and harmonic distortion estimates with and without a Kaiser window.

WINDOW	MEASURED DATA			SIMULATED DATA		
	$\hat{A}_m$	$\hat{\phi}_m$	100D PERCENT HARMONIC DISTORTION	100D PERCENT HARMONIC DISTORTION		
				0%	0.1%	1%
No Window	1.054	-123.3	0.060	0.000	0.100	1.000
- 30 dB Side Lobe	1.054	-123.3	0.071	0.032	0.073	0.972
- 40 dB Side Lobe	1.054	-123.3	0.077	0.018	0.084	0.984
- 50 dB Side Lobe	1.054	-123.3	0.082	0.009	0.092	0.992
- 60 dB Side Lobe	1.054	-123.3	0.087	0.004	0.097	0.997
- 70 dB Side Lobe	1.054	-123.3	0.097	0.002	0.099	0.999
- 80 dB Side Lobe	1.054	-123.3	0.093	0.001	0.099	0.999
- 90 dB Side Lobe	1.054	-123.3	0.095	0.000	0.100	1.000
-100 dB Side Lobe	1.054	-123.3	0.097	0.000	0.100	1.000

Computations performed on 20 cycles of data at 32 points per cycle. Measured data was the output of a synthesizer; simulated data was a computed sinewave with distortion added to the second harmonic as labeled.

When long sequences or long integration times are used to combat noise, the computation time can be significantly reduced by averaging. This efficiency is possible if the total sequence consists of two or more subsequences with an integer number of samples and cycles per subsequence. The simplest example is a data sequence of  $m$  cycles of the fundamental test sinusoid with each cycle containing  $n$  samples--this data sequence can be compressed from  $m$  cycles to one cycle by averaging. Both the SFDFT and windowed SFDFT computations can be speeded up through averaging. (The window is applied before averaging.)

#### Statistical Parameters

For signals which may be arbitrary periodic waveforms or noise-like data, statistical parameters are more appropriate in characterizing the data than are the SFDFT derived amplitude and phase (although naturally these parameters can be computed for sinusoidal signals as well).

The voltage extremes or range of the waveform are obtained by inspection of the data sequence, and both the maximum and minimum voltages are found.

The average or dc voltage is computed with

$$y_{dc} = \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i . \quad (12)$$

The standard deviation or true ac rms is obtained from

$$y_{rms} = \sigma_y = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2} . \quad (13)$$

The choice of factor  $1/n$  instead of  $1/(n-1)$  is to satisfy the Parseval relationship so that the rms amplitude of an ideal sine wave is the same whether estimated with the SFIDFT or with the true rms methods.

#### The second joint moment

$$E(xy) = \frac{1}{n} \sum_{i=1}^n x_i y_i , \quad (14)$$

is analogous to power if the data sequences  $x_i$  and  $y_i$  are proportional to current and voltage. Power measurement was the motivation for including this joint moment estimate. The total ac power,  $P_{ac}$ , is computed using

$$P_{ac} = E(xy) - \bar{x} \bar{y} . \quad (15)$$

All of the summations in Eqs. (10) through (15) can be interpreted as Euler-Maclaurin approximations to integrals of the continuous time functions. For functions that can be represented as a finite number of harmonically-related sinusoids, this simple approximation is exact [7].

#### SOFTWARE FUNCTIONS

The voltmeter software is organized around a monitor routine awaiting input from the host via the IEEE-488 interface (as shown in Fig. 8). The input consists of ASCII strings of variable length--the first two characters being a code for setup or control. Strings are terminated by an end-of-identify (EOI) sent with the last character.

"Setup" codes are accompanied by an encoded byte string specifying the value for a certain parameter, such as the programmable gain for a specific A/D converter. Setup strings tell the program what values are to be used, but do not perform any operation such as physically setting the gain of an A/D converter. No default parameters are present within the program; all parameters must be specifically loaded.

"Control" codes (two-byte strings) command the voltmeter to perform certain functions; i.e., make an A/D conversion, compute certain parameters, transfer data to the host, etc.

The voltmeter can be requested to compute and return various parameters as requested, or to return the original sampled waveform to the host for special processing not supported by the voltmeter (plotting, for example). It is necessary to point out that although commands are transferred to the voltmeter as ASCII byte strings, data is received from the voltmeter in PDP-11 format (two bytes forming the 16-bit integers for sampled data, or four bytes floating point variables for all computed parameters). This was done in the interest of faster data transfer and maintaining original data precision for NRL-USRD applications. This feature prevents full compatibility with non-PDP 11 host processors, although subroutines to encode integer and floating-point variables exist as part of the display functions, and the program could be modified accordingly.

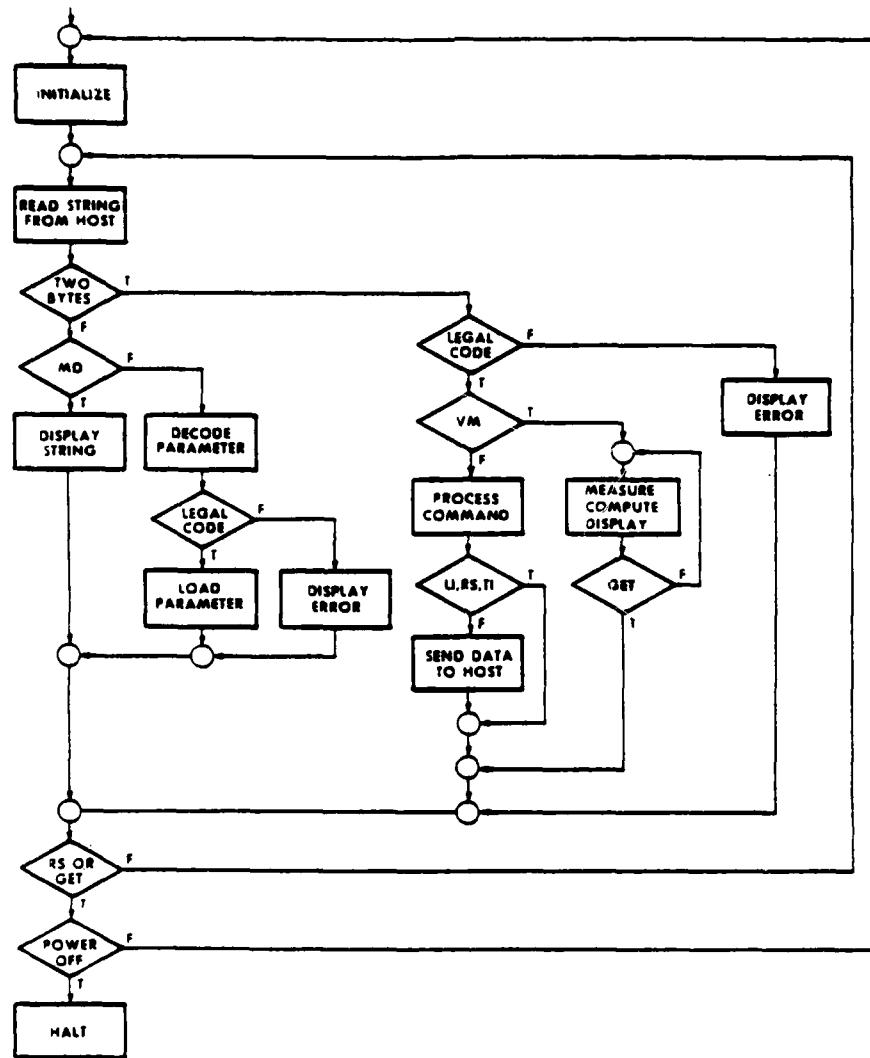


Fig. 9 - Flow chart of the basic monitor logic in the voltmeter software.

A special control code (VM) allows the voltmeter to measure, compute, and display as its own pace without host control; the host regains control by generating a group-execute-trigger (GET) on the IEEE-488 bus. The voltmeter displays data on either the console terminal or a 24-character plasma display. Display styles allowed are: count of A/D conversions and errors; voltage, phase, and distortion of one specific channel; peak voltage of all three channels; or voltages of two channels and the phase between the two.

The computation commands follow this sequence: the buffer of 12-bit digitized integer values is converted to floating-point numbers representing voltages within the A/D converter range ( $\pm 10$  V); a window function is applied, if desired; the data sequence in the buffer is averaged into a smaller subsequence, if desired; the specific computation is made; the results are scaled to account for the A/D converter gain and corrected for amplitude

change due to the window function, if used; and the data are transmitted to the host. The use of windowing and subsequence averaging seems not very useful for the computation of nonsinusoidal parameters.

The four primary commands are: "D" (compute voltage, phase, and distortion from SFDFT); its subset, "C" (the same without distortion), used for more rapid computation where distortion is not needed; "E" (total rms, peak, and dc voltage); and a similar subset, "F" (absolute peak only), a rapid tool for system autoranging or overload detection. Relative computation times are shown in Table 3. Note that these times refer to a PDP-11/23 computer with both the KEF-11 floating-point processor and the new FPF-11 floating-point hardware. The FPF-11 offers as much as a factor of 5 improvement in computation speed.

Table 3 - Sample computations measured for a PDP-11/23 with the KEF-11 and FPF-11 floating-point processors.

COMPUTATION STYLE	COMPUTATION TIME IN SECONDS							
	NO SUBSEQUENCE AVERAGING				WITH SUBSEQUENCE AVERAGING			
	NO WINDOW		WITH WINDOW		NO WINDOW		WITH WINDOW	
KEF-11	FPP-11	KEF-11	FPP-11	KEF-11	FPP-11	KEF-11	FPP-11	KEF-11
SINUSOIDAL PARAMETERS								
Voltage, Phase	0.494	0.149	0.570	0.172	0.210	0.098	0.293	0.132
Voltage, Phase, Distortion	2.627	0.506	2.632	0.531	0.383	0.133	0.458	0.165
NONSINUSOIDAL PARAMETERS								
rms, Peak, dc Voltage	0.350	0.151	-	-	0.198	0.100	-	-
Peak Voltage	0.189	0.096	-	-	-	-	-	-
Total Power	1.488	0.477	-	-	0.927	0.373	-	-

Average of 100 measurements of 20 cycles of data at 40 points per cycle (800 points). Distortion computed from first 7 harmonics. Subsequence averaging to 1 cycle. Times include data transfer on IEEE-488 bus (about 0.016 second).

Collateral computation commands include the ability to compute the total rms voltage and phase of a specific harmonic and to compute the power parameters second joint moment (ac and dc power) and covariance (ac power). The power parameters are the total power generated by all harmonics, and it is assumed that the host will scale the data properly to account for the calibration of the current measuring circuitry.

Further details of the software are to be found in the appendices and the liberally-commented source code located there. Appendix C lists various error and status messages; Appendix D gives listings for sample host routines to drive the voltmeter; and Appendix E includes complete listings of all software modules loaded into the voltmeter, with external discussion where appropriate.

#### SUMMARY

A multiple-channel microprocessor-based sampling voltmeter has been developed for use in the USRD measurement systems. This voltmeter is capable of measuring amplitude, phase, and distortion, and of using several computational algorithms to derive these results as the application requires. The original voltmeter has been used successfully in the Anechoic

Tank Facility as a high-voltage impedance measurement system, and this more powerful version has been greatly expanded to be used in Low-Frequency Facility, or other measurement systems as needed.

#### ACKNOWLEDGMENTS

The authors would like to acknowledge R.W. Anderson and R.W. Luckey, both of the USRD, and National Instruments, Inc., for their assistance in making this device function on the IEEE-488 interface bus, and C.K. Brown (USRD) for his assistance in incorporating the voltmeter into an operating electronic measurement system. This work was performed in the Acoustic Metrology Program sponsored by the Naval Sea Systems Command (SEA63R12).

#### REFERENCES

1. Barry A. Bell, Bruce F. Field, Thomas Kibalo, "A Fast Response Low-Frequency Sampling Voltmeter," National Bureau of Standards Tech Note 1159, National Bureau of Standards, Gaithersburg, MD, Aug 1982.
2. Dranetz Engineering Laboratories, Dranetz Model 3100 Sampling Network Analyzer Bulletin 3100, Edison, NJ.
3. G. Goertzel, "An Algorithm for the Evaluation of Finite Trig Series," *American Mathematical Monthly*, Vol. 65, 34-35 (1958).
4. G. Goertzel, *Mathematical Methods for Digital Computers, Vol I*, A. Ralston and H.S. Wolf, eds., John Wiley & Sons, New York, NY, 1960 (10th printing 1967), Ch. 24, 258-262.
5. F.J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proceedings of the IEEE*, Vol. 66, No. 1, 51-83 (1978).
6. J.F. Kaiser and R.W. Schafer, "On the Use of the  $I_0$ -Sinh Window for Spectrum Analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP-28, No. 1, 105-107 (1980).
7. Philip J. Davis and Philip Rabinowitz, *Numerical Integration*, Blaisdell Publishing Company, Waltham, Mass., 1967, Sec. 2.9.
8. National Instruments, "GPIB11V-1 Operating and Service Manual, April 1980, Part No. 320002-01; and "Software Reference Manual for National Instruments GPIB11-Series Interface Cards," September 1981, Part No. 310001-01.

## **APPENDIX A**

### **Voltmeter Hardware**

The hardware chosen as the heart of the voltmeter was the Digital Equipment Corporation (DEC) PDP-11/2 microcomputer (later upgraded to the PDP-11/23 with a KEF-11 floating-point processor), because it was a device that the USRD had both the experience and facilities to handle.

For A/D converters, ADAC Corp. models with DMA interfaces were chosen: model 1012, whose maximum sampling rate of 35 kHz was satisfactory for the LFF voltmeter, and model 1012AD, whose maximum sampling rate of 100 kHz was necessary for the ATF voltmeter.

Other interface modules used were:

MXV11-A	(DEC) Multifunction module: 2-serial line units 16K words RAM memory 4K words ROM memory (Intel 2732 EPROMs)
GPIBV11-1	(National Instruments) IEEE-488 bus interface
DRV11	(DEC) 16-bit parallel I/O card

The parallel I/O card is used only in pulsed systems to synchronize the measurements with the beginning of a toneburst.

(BLANK PAGE)

## APPENDIX B

### Voltmeter Commands

SETUP COMMANDS (# is channel number): These consist of a 2-byte code and a decimal integer string for the parameter value.

AE	Number of times to automatically retry a measurement if an A/D error occurs
AI	Which A/D-DMA device generates an interrupt when the data transfer is complete (1, 2, 3)
AN	Which A/D converters are to be used; additive, where 1 = #1, 2 = #2, 4 = #3, (7 = all three)
AV	Averaging mode (0 = no averaging, >0 = number of cycles in the averaged subsequence)
CT	Computation type (for future use)
DD	Display device: serial line 1 (console) or 2 (plasma display)
DS	Display style: 0 (conversion count), 1-3 (voltage, phase, distortion of channels 1-3), 4 (peak voltage in all 3 channels), 5 (voltages in channels 1-2 and phase between)
FC	Sample (clock) frequency in Hz
FS	Signal frequency in Hz
GV	Gated system (0 = not gated, 1 = use parallel I/O card to synchronize with toneburst)
G#	Programmable gain for A/D converter (1, 2, 5, 10)
HA	Number of harmonics used to compute distortion
HC	Which harmonic to compute (see B# - CONTROL COMMANDS)
MD	Transfer message to display, a byte string of up to 40 characters
NC	Number of cycles per sequence
PA	Number of points to convert per sequence
PR	Number of points per sequence to return to host

PS      First point to return to host. In practice, the first few  
 converted points are bad, therefore, PA = PR + PS points are  
 converted and PR points are computed  
  
 P#      Port (channel) for A/D Multiplexor (0-15)  
  
 WM      Window mode (0 = don't use window, 1 = use window)

**CONTROL COMMANDS (2-byte control codes; # is channel number)**

B#      Compute and return voltage and phase of the harmonic defined by HC  
 parameter  
  
 CW      Compute a sine wave for test purposes  
  
 C#      Compute and return voltage and phase of the fundamental  
  
 D#      Compute and return voltage, phase, and distortion of the  
 fundamental  
  
 E#      Compute and return statistical data (rms, dc, positive and  
 negative voltage extremes)  
  
 F#      Compute and return absolute peak voltage  
  
 LI      List setup parameters on the display  
  
 ME      Perform A/D conversion and return status  
  
 PO      Compute and return total power (second joint moment)  
  
 RP      Transfer setup parameters to host  
  
 RS      Clear conversion and error counters  
  
 R#      Read back a data sequence  
  
 RW      Read back the average of a window and the weighted sequence  
  
 SJ      Compute and return total power (second joint moment) and  
 covariance (ac power)  
  
 TI      Measure the time used in data transfer over the IEEE-488 bus  
  
 VM      Set to free-running display voltmeter mode  
  
 WI      Prepare voltmeter to read a computed window sequence from the host

## SPECIAL IEEE-488 COMMANDS

EOI End or identity - must coincide with last character sent by the host

GET Group execute trigger - command will interrupt whatever the voltmeter is doing and reinitialize

## APPENDIX C

### Voltmeter Status Messages

Status is returned as an integer variable that is the voltmeter's acknowledgment of a measure command. It refers to either the legality of the command or the success of executing it.

<u>STATUS</u>	<u>MEANING</u>
1	Normal return
-1	A/D error in channel 1
-2	A/D error in channel 2
-4	A/D error in channel 3
-10	DMA error in channel 1
-20	DMA error in channel 2
-40	DMA error in channel 3
-100	A/D setup error
-101	Too many points for program buffer

Errors -1 through -40 are added together in case of multiple errors, and thus cover all numbers between -1 and -77.

The voltmeter processor will halt for the following severe errors:

<u>ADDRESS</u>	<u>REASON FOR HALT</u>
002036	Floating-point exception (can proceed)
002054	Function (sine, atan, sqrt) error (can proceed)
003370	Too many samples to compute (can proceed)
006224	Illegal display mode (can proceed)
013236	IEEE-488 interface illegally strapped (fatal)

(BLANK PAGE)

**APPENDIX D**  
**Voltmeter Control Subroutines**

The following describes the protocol necessary and the available general-purpose subroutines for controlling this sampling voltmeter from a host processor. The National Instruments IEEE-488 bus driver and its software interface IBUP are a prerequisite [8].

Listings for the following subroutines (written at NRL-USRD) are included in this appendix.

**1. Primitive subroutines**

RDLSI	Poll the bus and if successful read a byte string from the voltmeter
WRLSI	Write a byte string to the voltmeter

**2. Higher-level I/O subroutines**

WRLPA	Send an integer parameter to voltmeter
WRLVM	Send an integer array to voltmeter
WRMEA	Send measurement command to voltmeter, wait an appropriate moment, enable specific clock synthesizer, receive status from voltmeter, disable specific clock synthesizer

**3. Example of voltmeter I/O**

METST	Test routine to command a measurement, read back a data sequence, and convert to volts
-------	--

```

0001      SUBROUTINE RDLSI (IADR,IBUF,ILEN,ISW)
C
C          SUBROUTINE TO READ A BYTE STRING FROM THE LSI VOLTMETER.
C
C          ARGUMENTS:
C              IADR = GPIB BUS ADDRESS
C              IBUF = BYTE-BUFFER TO LOAD WITH STRING
C              ILEN = LENGTH OF STRING EXPECTED
C              ISW = STATUS OF READ PROCESS
C                  ISW = 1 (SUCCESSFUL)
C
C          AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
C
0002      BYTE IBUF(ILEN)
0003      INTEGER GPIB
C
0004      100  CONTINUE
C
C          TEST FOR SRQ (CURRENTLY NOT WORKING WITH LSI)
C          J=GPIB(13)
CD        IF (J.NE.1) WRITE (6,110) J
CD110    FORMAT (' RDLSI TEST-SRQ FAILURE: '14)
C          IF (J.NE.1) GOTO 100
C
C          POLL FOR RESPONSE
0005      J=IBUP (6,IADR)
0006      IF (J.EQ.'102) GOTO 200 !SUCCESS
0007      IF (J.EQ.-6) GOTO 230 !TIMEOUT
D
D120    WRITE (6,120) J
FORMAT (' RDLSI POLL FAILURE: '14)
0008      GOTO 100           !TRY AGAIN
C
C          READ THE BYTE STRING
0009      200  J=IBUP (1,IADR,IBUF,ILEN)
0010      IF (J.EQ.ILEN) ISW=1      !SUCCESS
0011      IF (J.NE.ILEN) ISW=J      !FAILURE
D
D210    WRITE (6,210) J,ISW
FORMAT (' RDLSI STATUS: J='13' ISW='13)
D
D220    WRITE (6,220) IBUF
FORMAT (' RDLSI STRING='1204)
C
0012      RETURN
C
0013      230  CONTINUE
D
D240    WRITE (6,240) J
FORMAT (' RDLSI TIMEOUT: J='13),
0014      RETURN
C
0015      END

```

## PROGRAM SECTIONS

Name	Size	Attributes
*CODE1	000216	RW,I,CON,LCL
*PDATA	000010	RW,D,CON,LGL
*IDATA	000040	RW,D,CON,LCL
*VARS	000004	RW,D,CON,LCL

Total Space Allocated = 000272 93

No FPP Instructions Generated

FORTRAN IV-PLUS V3.0  
WRLSI.FTN:101

21:07:58 25-Mar-82  
/TR:BLOCKS/WR

Page 1

0001 SUBROUTINE WRLSI (IADR,STRING,ICNT,ISW)  
C  
C SUBROUTINE TO WRITE A BYTE STRING TO THE LSI VOLTMETER.  
C  
C ARGUMENTS:  
C IADR -- BUS ADDRESS OF DEVICE  
C STRING-- BYTE ARRAY TO BE SENT  
C ICNT -- LENGTH OF ARRAY TO BE SENT  
C ISW -- STATUS OF OPERATION  
C ISW=1 IS SUCCESSFUL  
C ISW=\* UNSUCCESSFUL (\* = GPIB ERROR)  
C  
C AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982  
C  
0002 BYTE STRING(ICNT)  
C  
0003 C  
C WRITE THE STRING  
J=IBUP(0,IADR,STRING,ICNT)  
C  
C DID IT WORK? (IF ISW = ICNT, SUCCESS).  
IF NOT, ISW IS GPIB ERROR STATUS.  
IF (J.NE.ICNT) ISW=J  
IF (J.EQ.ICNT) ISW=1  
C  
D WRITE (6,90) STRING  
D90 FORMAT (' WRLSI STRING='<ICNT>A1)  
D WRITE (6,100) IADR,J,ISW  
D100 FORMAT (' WRLSI STATUS: ADDR=''03' J=''13' ISW=''13)  
C  
0006 RETURN  
0007 END

#### PROGRAM SECTIONS

Name	Size	Attributes
\$CODE1	000134	46 RW,I,CON,LCL
\$PDATA	000004	2 RW,D,CON,LCL
\$IDATA	000032	13 RW,D,CON,LCL
SVARS	000002	1 RW,D,CON,LCL

Total Space Allocated = 000174 62

No FPP Instructions Generated

```

0001      SUBROUTINE WRLPA (IADR,ICODE,IVALUE,ISW)
C
C      SUBROUTINE TO WRITE A CODE AND AN INTEGER VALUE TO
C      THE LSI-VOLTMETER, AND READ AN ACKNOWLEDGEMENT.
C          IADR = GPIB BUS ADDRESS OF VOLTMETER
C          ICODE = TWO-BYTE PARAMETER SETUP CODE
C          IVALUE = POSITIVE INTEGER PARAMETER TO LOAD
C          ISW = STATUS OF OPERATION
C              ISW = 1 (SUCCESSFUL AS ACKNOWLEDGED)
C              ISW = * (FAILURE * AS ACKNOWLEDGED)
C
C      AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
C
0002      BYTE STRING(8)
0003      BYTE ICODE(2)
C
C      INCORPORATE THE CODE INTO THE STRING
0004      STRING(1)=ICODE(1)
0005      STRING(2)=ICODE(2)
C
C      ENCODE PARAMETER INTO SIX-CHARACTER ASCII STRING
0006      ENCODE (6,100,STRING(3)) IVALUE
0007      100 FORMAT (I6)
C
C      REPLACE LEADING SPACES WITH ZEROS
0008      DO 110 I=3,8
0009      110 IF (STRING(I).EQ.' ') STRING(I)='0'
C
0010      NB=8
C
C      WRITE THE STRING TO THE GPIB-BUS
D      WRITE (6,200) IVALUE,STRING
D200      FORMAT (' WRLPA VALUE='I6' STRING='8A1)
CALL WRLSI (IADR,STRING,NB,ISW)
0011      IF (ISW.NE.1) CALL TSTERR (6,ISW)      !ERROR IS ???
0012      IF (ISW.EQ.-6) J=IBUP(10)                !TIMEOUT CLEANUP
D      WRITE (6,300) ISW
D300      FORMAT (' WRLPA (WRITE) STATUS: ISW='I3)
C
C      READ REPLY IF GIVEN
CALL RDLSI (IADR,ISTAT,2,ISW)
C      IF (ISW.NE.1) CALL TSTERR (6,ISW)      !ERROR IS ???
C      IF (ISW.EQ.-6) J=IBUP(10)                !TIMEOUT CLEANUP
CD      WRITE (6,310) ISW
CD310      FORMAT (' WRLPA (READ) STATUS='I3)
C
C      SUBSTITUTE ACKNOWLEDGED STATUS FOR GPIB STATUS
C      (SW=ISTAT
0013      RETURN
0014      END

```

## PROGRAM SECTIONS

Name	Size	Attributes
\$CODE1	000252	85 RW,I,CON,LCL
\$PDATA	000010	4 RW,D,CON,LCL
\$IDATA	000032	13 RW,D,CON,LCL
\$VARS	000014	6 RW,D,CON,LCL

Total Space Allocated = 000330 108

No FPP Instructions Generated

```

0001      SUBROUTINE WRLVM (IADR,ICODE,IVAL,ICNT,ISW)
C
C      SUBROUTINE TO WRITE A BYTE STRING TO THE LSI-VOLTMETER
C      PREFIXED BY A TWO-BYTE PARAMETER CODE.
C          IADR = GPIB BUS ADDRESS
C          ICODE = TWO-BYTE PARAMETER CODE
C          IVAL = ARRAY TO SEND TO VOLTMETER
C          ICNT = NUMBER OF INTEGERS TO SEND
C          ISW = STATUS OF ACKNOWLEDGEMENT
C          IVAL COULD BE A 4-BYTE REAL VARIABLE (ICNT=2).
C          IVAL COULD BE NOTHING (ICNT=0; SEND CODE ONLY).
C
C          AUTHOR: R. E. SCOTT, JR. (NRL/USCD); VERSION: JANUARY 1982
C
0002      INTEGER ICODE
0003      INTEGER IVAL(8)
0004      INTEGER STRING(10)
C
C      PREFIX CODE INTO STRING
0005      STRING(1)=ICODE
0006      IF (ICNT.EQ.0) GOTO 60
C
C      ENTER VALUE INTO STRING, TWO BYTES AT A TIME
0007      DO 50 I=1,ICNT
0008      S0      STRING(I+1)=IVAL(I)
C
C      COMPUTE STRING LENGTH
0009      60      NB=2+2*ICNT
C
C      WRITE THE STRING
0010      CALL WRLSI (IADR,STRING,NB,ISW)
0011      IF (ISW.NE.1) CALL TSTERR (6,ISW)      !ERROR IS ???
C          IF (ISW.EQ.-6) J=IBUP(10)           !TIMEOUT CLEANUP
D        IF (ISW.NE.1) WRITE (6,100) ISW
D100    FORMAT (' WRLVM STATUS: ISW='13)
C
C      READ REPLY IF GIVEN
C      CALL RDLSI (IADR,ISTAT,2,ISW)
C          IF (ISW.NE.1) CALL TSTERR (6,ISW)      !ERROR IS ???
C          IF (ISW.EQ.-6) J=IBUP(10)           !TIMEOUT CLEANUP
CD        IF (ISW.NE.1) WRITE (6,110) ISW
CD110    FORMAT (' RDLSI STATUS: ISW='13)
C
C      EXCHANGE GPIB STATUS FOR ACKNOWLEDGEMENT STATUS
C      ISW=ISTAT
C
0012      RETURN
0013      END

```

## PROGRAM SECTIONS

Name	Size	Attributes
\$CODE1	000234	78 RW,I,CON,LCL
\$PDATA	000604	2 RW,D,CON,LCL
\$IDATA	000032	13 RW,D,CON,LCL
\$VARS	000030	12 RW,D,CON,LCL
\$TEMPS	000002	1 RW,D,CON,LCL

Total Space Allocated = 000324 106

No FPP Instructions Generated

```

0001      SUBROUTINE WRMEA (IADR,ICODE,FREQ,ITIME,ISTAT)
C
C      SUBROUTINE TO WRITE A "ME" (MEASURE) TO THE LSI-VOLTMETER,
C      (WAIT), ENABLE SAMPLING SYNTHESIZER, RECEIVE ACKNOWLEDGMENT,
C      AND DISABLE SAMPLING SYNTHESIZER.
C      *** SPECIFICALLY FOR LOW FREQUENCY SYSTEM SYNTHESIZER ***
C
C      IADR..... IEEE-488 BUS ADDRESS FOR VOLTMETER.
C      ICODE.... MEASUREMENT CODE (ME).
C      FREQ..... SAMPLING FREQUENCY IN HZ.
C      ITIME.... TIME TO WAIT BETWEEN MEASURE COMMAND AND ENABLING
C                  SAMPLE-RATE GENERATOR (MSEC).
C      ISTAT.... MEASUREMENT STATUS.
C
C      AUTHOR: R. E. SCOTT, JR. (NRL/USRD); VERSION: JANUARY 1982
C
0002      ISTAT=0          !STATUS: NO MEASUREMENT
C
C      COMMAND A MEASUREMENT
0003      CALL WRLSI (IADR,'ME',2,ISW)
0004      IF (ISW.NE.1) CALL TSTERR (6,ISW) !BUS ERROR?
0005      D100      IF (ISW.NE.1) WRITE (6,100) ISW
                  FORMAT (' WRMEA WRITE-"ME" STATUS: ',13)
0006      C         IF (ISW.NE.1) RETURN           !GIVE UP IF BUS ERROR
0007      CALL WAIT (ITIME,I,K)          !WAIT FOR SLAVE TO BE READY
0008      CALL SM102 ("24,FREQ,'E ',JSW,ISW)!ENABLE SYNTHESIZER (CLOCK)
0009      CALL RDLSI (IADR,ISTAT,2,ISW)  !GET STATUS FROM SLAVE
0010      D110      IF (ISW.NE.1) CALL TSTERR (6,ISW) !BUS ERROR?
                  IF (ISW.NE.1) WRITE (6,110) ISW
                  FORMAT (' WRMEA ACKNOWLEDGE STATUS: ',13)
0011      CALL SM102 ("24,FREQ,'D ',JSW,ISW)!DISABLE SYNTHESIZER AGAIN
0012      C         RETURN
END

```

## PROGRAM SECTIONS

Name	Size	Attributes
SCODE1	000252	RW,I,CON,LCL
SPDATA	000042	RW,D,CON,LCL
SIDATA	000064	RW,D,CON,LCL
SVARS	000006	RW,D,CON,LCL

Total Space Allocated = 000406 131

No FPP Instructions Generated

```

0001      SUBROUTINE METST (IAD,IT,MODE)

C      SUBROUTINE TO TEST THE LEESBURG/RD-11/LSI-11/A22D
C      HIGH-POWER MEASUREMENT SYSTEM INPUTS.

0002      INCLUDE 'LSPARM.COM'
0003 *      COMMON /LSPARM/SIFREQ,SIAMPL,SAFREQ,SAAMPL,
*      ICAIN(3),NAD,NADT,NPAD,NPTR,NPST,NCYC,NHAR,
*      2AVG,1DST,ICTYP,IAERS,IGATE,1WINDO,1DLUN,1HARC,
*      3IPORT(3)
*      C      SIFREQ,SIAMPL----SIGNAL FREQUENCY AND AMPLITUDE
*      SAFREQ,SAAMPL----SAMPLE FREQUENCY AND AMPLITUDE
*      ICAIN----A/D PROGRAMMABLE GAIN
*      NAD----WHICH A/D'S USED (I=1,2=2,3=4,ALL=7)
*      NADT----WHICH A/D (DMA) TO INTERRUPT ON
*      NPAD----NUMBER OF POINTS TO CONVERT
*      NPTR----NUMBER OF POINTS TO TRANSFER
*      NPST----FIRST POINT TO TRANSFER
*      NCYC----NUMBER OF CYCLES TO CONVERT
*      NHAR----NUMBER OF HARMONICS USED TO COMPUTE DISTORTION
*      2AVG----NUMBER OF CYCLES TO AVERAGE TO (0=DON'T)
*      1DST----DISPLAY STYLE (0~5)
*      ICTYP----COMPUTATION TYPE
*      IAERS----A/D ERRORS TO RETRY BEFORE QUITTING
*      IGATE----GATED (1) OR NOT (0) ((DRV-11 IN USE))
*      1WINDO----MULTIPLY DATA BY WINDOW (WDATA)
*      1DLUN----DISPLAY DEVICE
*      1HARC----HARMONIC TO COMPUTE
*      C
0004      INCLUDE 'LSDATA.COM'
0005 *      PARAMETER BSIZE=1024
0006 *      PARAMETER NOAD=3
0007 *      COMMON /1DAT/1DATA(NOAD*BSIZE)
0008 *      COMMON /RDAT/RDATA(NOAD*BSIZE)
0009 *      COMMON /WDAT/WDATA(BSIZE),WATT,WAV
0010 *      COMMON /SDAT/SDATA(BSIZE)
0011 *      COMMON /RTIM/TIM(NOAD)
*      C      MAXIMUM POINTS TO READ
*      C      NUMBER OF CHANNELS
*      C      RAW A/D DATA (IN BITS)
*      C      NORMALIZED DATA (IN VOLTS)
*      C      WINDOW, ATTENUATION, AVERAGE
*      C      COMPUTATION BUFFER
*      C      COMPUTATION TIMES
0012      INTEGER (PAR(15),IAE(6))

0013      IF (IT.EQ.'ME') GOTO 100
0014      IF (IT.EQ.'RI') GOTO 200
0015      IF (IT.EQ.'R2') GOTO 260
0016      IF (IT.EQ.'R3') GOTO 260
C      C      GOTO TO LS1 TO PERFORM A/D CONVERSION (ME)
0017      100      CONTINUE
C      C      WRITE (6,101)
C      C      101      FORMAT (' DO A/D CONVERSION')
0018      IF (MODE.NE.'LO') CALL WRLVM (IAD,'ME',0,0,ISW)
0019      IF (MODE.EQ.'LO') CALL WRMEA (IAD,'ME',SAFREQ,200,ISW)
0020      IF (ISW.GE.0) RETURN
0021      WRITE (6,102) ISW
0022      102      FORMAT (' *** A/D CONVERSION ERROR: '13' ***')
0023      IF (ISW.EQ.-100) WRITE (6,110)
0024      IF (ISW.EQ.-101) WRITE (6,120)
0025      110      FORMAT (' A/D SETUP ERROR (A/D'S OR INTERRUPT)')
0026      120      FORMAT (' TOO MANY POINTS REQUESTED')
0027      RETURN
C      C      READ A DATA BUFFER FROM THE LSI (R1,R2,R3)
0028      200      CONTINUE
C      C      WRITE (6,201)
C      C      201      FORMAT (' READ FROM 11/03')
0029      RDS=SECONDS(0,0)
0030      CALL WRLVM (IAD,IT,0,0,ISW)

```

```

0031      IF (ISW.NE.1) WRITE (6,203) ISW
0032 203  FORMAT (' WRLVM ERROR: '13)
0033      IF (ISW.EQ.-6) J=IBUP(10)
0034      IF (ISW.NE.1) RETURN
0035      L (IT.EQ.'R1') J=1
0036      IF (IT.EQ.'R2') J=2
0037      IF (IT.EQ.'R3') J=3
0038      K=(J-1)*NPTR+1
0039      CALL RDLSI (IAD,1DATA(K),2*NPTR,ISW)
0040      C
0041      C220  WRITE (6,220) (IDATA(I),I=K,K+NPTR)
0042      FORMAT (' '20B)
0043      IF (ISW.NE.1) WRITE (6,230) J,ISW
0044      230  FORMAT (' *** A/D DATA CHANNEL ''11'' IN ERROR: '13' ***')
0045      IF (ISW.EQ.-6) J=IBUP(10)
0046      IF (ISW.NE.1) RETURN
0047      C
0048      C
0049      FIX THE DATA PROPERLY
0050      L1=1+(J-1)*NPTR
0051      L2=L1+NPTR
0052      D0 250 I=L1,L2
0053      IM=1AND("7777",IDATA(I))
0054      IF (IM.GE."4000") IM=1OR("170000",IM)
0055      RDATA(I)=10.0*IM/"4000"
0056      CONTINUE
0057      TIM(J)=SECONDS(RDS)
0058      D
0059      IF ((IT.EQ.'R1').OR.(IT.EQ.'R2')) WRITE (6,260) TIM(J)
0060      D
0061      IF ((IT.EQ.'R3').OR.(IT.EQ.'RE')) WRITE (6,260) TIM(J)
0062      D260  FORMAT (' BUFFER READ TIME='FB.3)
0063      C
0064      RETURN
0065      END

```

## PROGRAM SECTIONS

Name	Size	Attributes
\$CODE1	001156	311 RW,I,CON,LCL
\$PDATA	000324	196 RW,D,CON,LCL
\$1DATA	000056	23 RW,D,CON,LCL
\$VARS	000074	36 RW,D,CON,LCL
LSPARM	000072	29 RW,D,OVR,GBL
1DAT	014000	3072 RW,D,OVR,GBL
RDAT	030000	6144 RW,D,OVR,GBL
WDAT	010010	2052 RW,D,OVR,GBL
SDAT	010000	2048 RW,D,OVR,GBL
ITIM	000014	6 RW,D,OVR,GBL

Total Space Allocated = 063772 13821

## APPENDIX E

### Voltmeter Program Listings

The modules used in this program are written in PDP-11 assembly language, divided into pure (ROM) code (modules 1-7) with all dynamic variables (RAM) in the final module.

- (1) LSIVM Command interpreter; measurement, computation, and display driver
- (2) AD2D A/D converter control subroutine
- (3) CONVRT Floating to ASCII encoding subroutine, used to prepare numbers for display or transfer to the host
- (4) DAPD Computation of amplitude, phase, distortion; uses DGZL below to obtain real and imaginary parts of voltage
- (5) DGZL Single frequency DFT subroutine: computes real and imaginary parts of voltage of a given spectral line by Goertzel's algorithm
- (6) GPIOSR IEEE-488 bus input and output subroutines, initialize subroutine, and interrupt service routine
- (7) GRMS Computation by time-integration of rms, dc, positive peak, voltages
- (8) LSIVAR All the dynamic variables used by (1)-(6) above

Also used are modules from DEC's RSX-11M System Library--\$ATAN (arctangent), \$SIN (sine), and \$SQRT (square root).

The subroutines (2)-(7) are written to conform with DEC standards; argument lists are passed through register 5 so that these routines could be used by other programs. Conditional assembly parameters allow dynamic variables to be assembled locally, or in an external global file such as LSIVAR.

(1) MODULE LSIVM - Main program for controlling the microcomputer-controlled sampling voltmeter

Internal Summary:

<u>(LABEL)</u>	<u>(FUNCTION)</u>	<u>(PAGE)</u>
<u>MONITOR ROUTINES</u>		
START	Initialize system	5
RDCODE	Read string from bus; interpret type of command	5
<u>PARAMETER ROUTINES</u>		
PARAM	Parameter decode and load	6
FREQP	Read frequency data	8
MESDAY	Read and display text string	8
DECOD	Byte string to integer decoding	9
FPISR	Floating-point processor interrupt service routine	9
EXEC	Interpret and direct control codes	10
CERR	Parameter I/O error routine	11
<u>CONTROL ROUTINES</u>		
ABORT	Abort process and reinitialize	11
MESURE	Make measurement	12
WAKE	Counter/display initialization	12
PARLST	Display setup parameters	13
IFWAIT	Delay subroutine	13
PARSND	Send setup parameters to host	14
ADSEND	Send A/D buffer or window to host	14
<u>PRE-PROCESSING ROUTINES</u>		
CMPUT	Computation interpreter	15
CWAVE	Compute test sine wave	17
FLOT	Convert A/D buffer to floating point buffer	18

<u>(LABEL)</u>	<u>(FUNCTION)</u>	<u>(PAGE)</u>
WINDIN	Read sequence window from host; compute average	19
WINDOW	Multiply data buffer by window buffer	19
BAVER	Average data buffer to fewer cycles	20
<u>COMPUTATION ROUTINES</u>		
GDFTD	Compute voltage, phase, distortion (DFT) of fundamental	21
GDFT	Compute voltage, phase (DFT) of fundamental	21
GSL	Compute voltage, phase (DFT) of harmonic	21
TDFTD	Transfer voltage, phase, distortion to host	22
TDFT	Transfer voltage, phase to host; display	22
DSDL	Load DFT values for display	22
GRMSV	Do statistical computation	23
GPEKO	Find absolute peak value	23
TRMSV	Transfer statistical data to host; display	24
TPEK	Transfer peak voltage to host; display	24
DEGAIN	Correct voltage for A/D gain	24
REWIND	Correct DFT voltage for window average	24
GPWR	Compute total power (second joint moment)	25
SJC	Compute total power and ac power	26
GMN	Get rms voltage for power computation	26
TJC	Transfer total power and ac power to host	26
<u>DISPLAY ROUTINES</u>		
VMDISP	Stand-alone voltmeter executive	27
CHODIS	Display, mode 0 (conversion and error count)	29
CH1DIS	Display, modes 1-3 (voltage, phase, and distortion)	30

CH4DIS	Display, mode 4 (voltages)	32
CH5DIS	Display, mode 5 (voltages and relative phase)	33
WRITE	Terminal output (string)	34
PRINT	Terminal output (integer)	34

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51

5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51

NRL-USRD MICROCOMPUTER SAMPLING VOLTmeter PROGRAM.

AUTHOR: RICHARD E. SCOTT, JR. (FEBRUARY 1982)

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51

THIS PROGRAM IS DESIGNED TO WORK AS A SAMPLING VOLTMETER CONTROLLED BY THE GPIB INTERFACE BUS. THE SETUP AND CONTROL COMMANDS ARE SENT IN MULTI-DYTE PACKETS; DATA IS RETURNED ON COMMAND IN THE FORM OF A WAVEFORM DIGITIZED BY UP TO THREE A/D CONVERTERS, AND AS VARIOUS COMPUTED PARAMETERS.

THE "MEASURE" COMMAND IS EFFECTIVELY THE ONLY ONE THAT IS ACKNOWLEDGED BY THIS PROGRAM, RETURNING AN INTEGER STATUS.

THE CURRENT VERSION CAN RETURN THE FOLLOWING PARAMETERS UNDER HOST CONTROL: RMS, PEAK, AND DC VOLTAGE FROM TIME INTEGRATION; RMS VOLTAGE, PHASE, AND DISTORTION FROM A FFT CALCULATION. ALSO RETURNED IS TOTAL POWER (POWER IN ALL HARMONICS). PROCESSING OPTIONS INCLUDE MULTIPLYING THE DATA SEQUENCE BY A WINDOW SPECIFIED BY THE HOST; AND AVERAGING MANY CYCLED SEQUENCES DOWN TO FEW CYCLES WHERE SUCH IS PERMISSABLE.

THE PROGRAM CONTAINS SEPARATE, CONTINUOUS DATA BUFFERS OF A GIVEN MAXIMUM SIZE FOR EACH OF THE THREE CHANNELS. HOWEVER, DATA IS DIGITIZED INTO MEMORY AS IF THESE ARRAYS WERE CONTIGUOUS, BUT SIZED FOR THE NUMBER OF SAMPLES DIGITIZED. THE COMPUTATION IS PERFORMED ON ONE CHANNEL'S DATA AT A TIME, AS EACH INTEGER SEQUENCE MUST FIRST BE CONVERTED TO FLOATING-POINT VALUES STORED IN THE SINGLE "TEMPORARY" FLOATING ARRAY. POWER, REQUIRING TWO FLOATING-POINT CHANNELS SIMULTANEOUSLY, IS COMPUTED DIFFERENTLY (A SAMPLE AT A TIME).

ALTHOUGH THIS DEVICE CAN ACT LIKE A DISPLAY VOLTMETER INDEPENDENT OF COMPUTER CONTROL, IT IS STILL NECESSARY TO SET UP THE DEVICE UNDER COMPUTER/BUS CONTROL; BECAUSE IT IS FREQUENCY-DEPENDENT, NO DEFAULT VALUES ARE PRACTICAL. TO ALLOW IT TO POWER-ON INTO A VOLTAGE-READING MODE.

THE VOLTMETER IS CONTROLLED BY COMMANDS FROM THE BUS, WHICH CONSIST OF A TWO-BYTE (TWO CHARACTER) CODE, AND AN INTEGER ARGUMENT (IF NEEDED) AS A BYTE STRING.

ANY PROCESS CAN BE PERMANENTLY ABORTED FROM THE HOST BY TRANSMITTING A DEVICE TRIGGER (GET) ON THE BUS.

ALL DEVICES AND COUNTERS ARE RESET.

```

53 :SETUP COMMANDS (TWO BYTES PLUS BYTE STRING)-
54 :FORMAT (TWO-BYTE CODE) (MULTI-BYTE VALUE)
55 :G: PROGRAMMABLE GAIN FOR A/D CONVERTER # (1-3) (1,2,5,10)
56 :P: INPUT PORT FOR A/D CONVERTER # (1-3) (0-16)
57 :A: WHICH OF A/D'S USED (#=1, #2=2, #3=4; ALL=7)
58 :AI: WHICH A/D TO INTERRUPT ON (DMA FINISH)
59 :AE: HOW MANY A/D ERRORS TO RETRY BEFORE GIVEUP
60 :PA: NUMBER OF POINTS (CONVERSIONS) PER WAVEFORM SEQUENCE
61 :PR: NUMBER OF POINTS (CONVERSIONS) TO RETURN TO HOST
62 :PS: FIRST POINT (CONVERSION) NUMBER TO RETURN TO HOST
63 :NC: NUMBER OF CYCLES PER SEQUENCE
64 :HA: NUMBER OF HARMONICS TO USE COMPUTING DISTORTION
65 :HC: WHICH HARMONIC TO COMPUTE
66 :DS: DISPLAY TYPE (Q.V.)
67 :DD: DISPLAY DEVICE (SEE PAGE 3)
68 :AV: AVERAGING MODE (0=NONE, N=CYCLES RESULTING)
69 :CT: COMPUTATION TYPE (Q.V.)
70 :GV: GATED (0=NO, 1=YES)
71 :WN: WINDOW MODE (0=NONE, 1=USE)
72 :MD: SEND MESSAGE TO DISPLAY
73
74
75 :INFORMATIONAL COMMANDS (SIX BYTES)-(OBSOLETE)
76 :FORMAT (TWO-BYTE CODE) (FOUR-BYTE REAL VALUE)
77 :FS: SIGNAL FREQUENCY IN HZ
78 :FC: CLOCK FREQUENCY IN HZ
79
80
81 :CONTROL COMMANDS (TWO BYTES)-
82 :FORMAT: (TWO-BYTE CODE)
83 :RS: CLEAR COUNTERS
84 :ME: PERFORM A/D CONVERSION AS PROGRAMMED
85 :LI: LIST 17 PARAMETERS ON THE LSIVM CONSOLE
86 :VM: GO TO VOLTMETER MODE (AUTOMATIC)
87 :WI: INPUT WINDOW DATA ("PR" REAL VALUES, 4*PR BYTES)
88 :TI: DOWRY COMMAND FOR 1/O TIMING INFORMATION
89 :CW: COMPUTE SINE WAVEFORM FOR TIMING INFORMATION
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
:COMPUTE COMMANDS (TWO BYTES)-
:FORMAT: (TWO-BYTE CODE)
:B: RETURN VOLTAGE AND PHASE OF NUTH HARMONIC, CHANNEL # (1-3)
:C: RETURN VOLTAGE AND PHASE OF FUNDAMENTAL, CHANNEL # (1-3)
:D: RETURN VOLTAGE, PHASE, DISTORTION (FROM DFT), CHANNEL # (1-3)
:E: RETURN RMS, PEAK+, PEAK-, DC VOLTAGES, CHANNEL # (1-3)
:F: RETURN PEAK ONLY, CHANNEL # (1-3)
:P: RETURN TOTAL POWER FROM CHANNEL #1 (VOLTS) AND #2 (AMPS)
:CJ: RETURN SECOND JOINT MOMENT AND COVARIANCE, CHANNELS #1 AND #2

```

```

110 ;SUBROUTINES RECALLED
111
112 :GP10SR -- GPIP-43B RMS CONTROL SUBROUTINES
113 :A32B -- ADAC A/D CONVERTER CONTROL SUBROUTINE
114 :CONVRT -- READ TO AND FROM A/D FOR DISPLAY
115 :GDIS -- COMPUTE TWO-SQUARES VOLTMAGES
116 :DCZL -- COMPUTE DC-ZEROES FOR PHOTOTRANSISTOR
117 :DAVD -- COMPUTE DATA VOLTAGE PHASE AND DISTANCE
118 :DVAR -- GLOBAL DATA VARIABLE
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

129 ;SPECIAL ASSEMBLY INSTRUCTION GRS3:
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

129 ;THIS SYSTEM WAS DESIGNED TO BE LOADED INTO PAGE 1000, SO THE
130 ;PAGE VARIABLES ARE CONCENTRATED IN THE MODULE LSIVAR. IT
131 ;IS NECESSARY TO CHECK THE STATUS OF THE CONDITIONAL
132 ;ASSIGNMENT PARAMETER "PAGE" IN THE MAIN ROUTINE LSIVM AND
133 ;MANY OF THE SUBROUTINES TO RESTORE THAT IT IS IN THE RIGHT
134 ;STATE. IN LSIVM PAGE 2000 CHECKS WHETHER PAGE 2000 CVERCR
135 ;IS IN PAGE 2000 (NOT 1000). IN A32D PAGE 2000 CVERCR
136 ;REFERS TO PAGE 2000. PAGE 2000 IS NOT LOADED, IN BEZL AND
137 ;DAPL IT CONCERN'S WHETHER PAGE 2000 VARIABLES ARE LOCALITY ASSEMBLED
138 ;(AS FOR OTHER USERS), OR IN LSIVAR THIS ASSEMBRATION.
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

129 ;SPECIAL TASK-BUILD INSTRUCTION 013. NOTE THAT THIS TASK IS
130 ;DESIGNED ONLY TO RUN ON A ONE-TO-ONE PAGE-1000 AND WILL NOT
131 ;EXECUTE UNDER A NEW-PAGE-1000 ASSET RELOCATABLE.
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

129 ;GR3 COMMAND FILE:
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

129 ;$IVM/SO/-UN/FP/LSV1000.GRP=1000
130 ;GR3,A32D,CONVRT,BEZL,DCZL
131 ;GP10SR
132 ;11.LSYSLIB.OBJLIB:MAIN:$SIN:EDSN:EGS1C
133 ;LSIVAR
134
135 ;SLACK=0
136 ;UNITS=9
137 ;PAR=PAR:1000:100001
138 ;ACTFL1=4
139 ;MAXFL1=8
140 ;WPFL1=6
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

129 ;AC0=70
130 ;AC1=21
131 ;AC2=22
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158

```

MACRO M1113 23-JUL-03 14:12 FACT 4

```

215
216
217
218 ; INITIALIZE WHATEVER IS NECESSARY AND GO INTO "WAIT" STATE.
219 ; NORMAL STATUS IS TO WAIT FOR INPUT FROM THE IEEE-488 BUS;
; WHEN RECEIVED, GO TO EITHER DATA OR COMMAND INTERPRETER.
220
221 001096 012766 000000C
222 001004 000005
223 001004 000005
224
225
226
227
228
229
230 001096 005037 000000C
231 001012 005037 000000C
232 001016 012763 001112
233 001022 004767 000000C
234 001026 012766 000000
235 001032 106400
236
237
238
239
240
241 001034 012767 000001 000000C RCODE:
242 001034 012705 001102
243 001042 012705 001102
244 001046 004767 000000C
245
246 001052 023727 000000C 0000002
247 001060 001002 0000002
248 001062 000167 000772
249
250 001066 022737 0042115 000000C 100:
251 001074 001012 000000C
252 001076 000626 000000
253
254 001092 0000003 001116' GPIARG: 3,1NAUF,RLEN,1BLFN
255 001110 000000G 002410'
256 001112 000001 000050
257 001116 000000G

```

;INITIALIZE STACK POINT'R  
;CLEAR Q-BUS

START: . F0V ;STACK,SP  
 RESET PROM ;LOAD FLOATING-ERROR INTERRUPT VECTOR  
 .IF EQ #FPISR,0\*244 ;AND PSW  
 MOV #349,0\*246 ;LOAD FUNCTION-ERROR INTERRUPT VECTOR  
 MOV #FOISR,0\*34 ;AND PSW  
 MOV #340,0\*36  
 .ENDC CLR @\*1CA  
 CLR @\*1CE  
 MOV #CPINA,R5  
 PC,GPIN1  
 JSR #0,RO  
 MOV R0,RTPS  
 ;NUMBER OF A/D CONVERSIONS  
 ;NUMBER OF A/D ERRORS  
 ;INITIALIZE IEEE-488 BUS CARD  
 ;CLEAR PSW  
 ;ENABLE INTERRUPTS

THIS PORTION OF THE PROGRAM AWAITS A DATA PACKET FROM THE  
 HOST PROCESSOR AND DIRECTS IT TO THE PROPER FUNCTION.

;DEFUALT STATUS = 1  
 ;READ A CODE FROM THE GPIB  
 ;HOW MANY BYTES?  
 ;IF TWO, EXECUTE A COMMAND  
 ;OTHERWISE INTERPRET DATA  
 ;CODE = "RD" ?  
 ;NO  
 ;YES, REQUIRES SPECIAL PROCESSING

;MAXIMUM INPUT STRING LENGTH  
;MAXIMUM DATA BUFFER SIZE

259  
 260  
 261  
 262  
 263  
 264 001122 004767 000634  
 265 001122 004767 000634  
 266 001126 022737 030507 000000G  
 267 001126 022737 030507 000000G  
 268 001134 001004 001004  
 269 001136 010337 000000G  
 270 001142 000167 177666  
 271 001146 022737 031167 000000G 103:  
 272 001146 022737 031167 000000G 103:  
 273 001154 001004 001004  
 274 001156 010337 000000G  
 275 001162 000167 177646  
 276 001166 022737 031507 000000G 203:  
 277 001166 022737 031507 000000G 203:  
 278 001174 001004 001004  
 279 001176 010337 000000G  
 280 001202 000167 177626  
 281 001206 022737 047101 0000000C 303:  
 282 001214 001004 001004  
 283 001216 010337 000000G  
 284 001222 000167 177606  
 285 001226 022737 044501 0000000C 403:  
 286 001234 001004 001004  
 287 001236 010337 000000G  
 288 001234 001004 001004  
 289 001234 001004 001004  
 290 001242 000167 177566  
 291 001246 022737 040520 0000000C 503:  
 292 001254 001004 001004  
 293 001256 010337 000000G  
 294 001256 010337 000000G  
 295 001262 000167 177546  
 296 001266 022737 051120 0000000C 603:  
 297 001274 001004 001004  
 298 001276 010337 000000G  
 299 001302 000167 177326  
 300 001306 022737 051520 0000000C 703:  
 301 001306 022737 051520 0000000C 703:  
 302 001306 022737 051520 0000000C 703:  
 303 001314 001004 001004  
 304 001316 010337 000000G  
 305 001322 000167 177506  
 306 001326 022737 041516 0000000C 803:  
 307 001334 001004 001004  
 308 001334 001004 001004  
 309 001336 010337 000000G  
 310 001342 000167 177466  
 311 001346 022737 040510 0000000C 903:  
 312 001354 001003 001003  
 313 001356 010337 000000G  
 314 001362 000400 000400

; THIS PORTION OF THE PROGRAM HANDLES THE INPUT OF DATA  
 ; PARAMETERS RECEIVED BYT STRING IS DECODED INTO AN  
 ; INTEGER VARIABLE IN R3, AND THEN LOADED INTO THE VARIABLE  
 ; SPECIFIED BY THE CODE WORD IC.

PARAM:	JSR	PC, DECODE	:DECODE ASCII STRING INTO BINARY INTEGER
266	CMP BNE NOV JMP	"G1", \$1C 108 R3, \$, GAIN1 RD CODE	:CODE = "G1"?
271	CMP BNE NOV JMP	"G2", \$1C 208 R3, \$, GAIN2 RD CODE	:CODE = "G2"?
276	CMP BNE NOV JHP	"G3", \$1C 308 R3, \$, GAIN3 RD CODE	:CODE = "G3"?
281	CMP BNE NOV JHP	"AN", \$1C 408 R3, \$, ADS RD CODE	:CODE = "AN"?
286	CMP BNE NOV JHP	"AI", \$1C 508 R3, \$, A/D RD CODE	:CODE = "AI"?
291	CMP BNE NOV JHP	"PA", \$1C 608 R3, \$, PAD RD CODE	:CODE = "PA"?
296	CMP BNE NOV JHP	"PR", \$1C 708 R3, \$, PTR RD CODE	:CODE = "PR"?
301	CMP BNE NOV JHP	"PS", \$1C 808 R3, \$, SHIFT RD CODE	:CODE = "PS"?
306	CMP BNE NOV JHP	"NC", \$1C 908 R3, \$, NCYC RD CODE	:CODE = "NC"?
309	CMP BNE NOV JHP	"HA", \$1C PARK R3, \$, MAR RD CODE	:CODE = "HA"?
310	CMP BNE NOV BR	"MAR", \$1C PARK R3, \$, MAR RD CODE	:HOW MANY HARMONICS TO PROCESS

; NUMBER SAMPLES TO RETURN TO HOST

; FIRST SAMPLE TO RETURN TO HOST

; NUMBER CYCLES PER SEQUENCE

```

317 001364 22737 053161 909069G PARY : CMP :AVG,@@1C :CODE = "AV"?
318 001372 001004 010337 060090C RNE 20S :RC,@@AVGOD :AVERAGING PROCESS TO FOLLOW
319 001374 010337 060090C NOV RDCode
320 001406 000167 177410 JNP
321
322 001404 022737 051594 000000G 268 : CMP :DS,@@1C :CODE = "DS"?
323 001412 001004 010337 060090C RNE 30S :RC,@@DSTYLE :DISPLAY STYLE
324 001414 010337 060090C LGA RDCode
325 001426 000167 177410 JNP
326
327 001424 022737 051597 000000G 362 : CMP :CV,@@1C :CODE = "CV"?
328 001432 001004 010337 060090C BNE 40S :RC,@@EGATED :GATING FLAG
329 001434 010337 060090C NOV RDCode
330 001436 000167 177410 JNP
331
332 001444 022737 042501 000000G 408 : CMP :AE,@@1C :CODE = "AE"?
333 001452 001004 010337 060090C BNE 50S :RC,@@AFERS :RETRY ERRORS COUNTFR
334 001454 010337 060090C NOV RDCode
335 001460 000167 177350 JNP
336
337 001464 022737 052153 000000G 508 : CMP :CT,@@1C :CODE = "CT"?
338 001472 001004 010337 060090C BNE 60S :RC,@@COMTYP :COMPUTATION TYPE
339 001474 010337 060090C NOV RDCode
340 001500 000167 177330 JNP
341
342 001504 022737 042104 000000G 608 : CMP :DB,@@1C :CODE = "DB"?
343 001512 001004 010337 060090C BNE 70S :RC,@@DLUN :DISPLAY LUN
344 001514 010337 060090C NOV RDCode
345 001520 000167 177310 JNP
346
347 001524 022737 046527 000000G 708 : CMP :WH,@@1C :CODE = "WH"?
348 001532 001004 010337 060090C BNE 80S :RC,@@WMODE :WINDOW MODE
349 001534 010337 060090C NOV RDCode
350 001540 000167 177270 JNP
351
352 001544 022737 041519 000000G 809 : CMP :HC,@@1C :CODE = "HC"?
353 001552 001004 010337 060090C BNE 90S :RC,@@AD1C :HARMONIC TO COMPUTE
354 001554 010337 060090C NOV RDCode
355 001560 000167 177259 JNP
356
357 001564 022737 0309520 000000G 903 : CMP :PI,@@1C :CODE = "PI"?
358 001572 001004 010337 060090C BNE 100S :RC,@@AD2C :A, D CHANNEL 1 PORT
359 001574 010337 060090C NOV RDCode
360 001600 000167 177210 JNP
361
362 001604 022737 0311129 000000G 1008 : CMP :P2,@@1C :CODE = "P2"?
363 001612 001004 010337 060090C BNE 110S :RC,@@AD2C :A, D CHANNEL 2 PORT
364 001614 010337 060090C NOV RDCode
365 001620 000167 177230 JNP
366
367 001624 022737 0311520 000000G 1108 : CMP :P3,@@1C :CODE = "P3"?
368 001632 001002 010337 060090G BNE 120S :RC,@@AD3C :A/L CHANNEL 3 PORT
369 001634 010337 060090G NOV RDCode
370 001640 000167 177170 1208 : JNP
371

```

373  
 374  
 375  
 376  
 377  
 378 001644 012737 051506 0.500 0.00:  
 379 001644 012737 051506 0.500 0.00:  
 380 001652 011097 000006C 0000069:  
 381 001654 013737 000002C 0000026:  
 382 001662 013737 000002C 0000026:  
 383 001670 000115:  
 384  
 385 001672 022737 041506 006900G 103:  
 386 001709 001067  
 387 001702 013737 000000G 000000G  
 388 001710 013737 000002C 000002C  
 389 001716 000462:  
 390  
 391  
 392 001720 000167 000400 20\$:  
 393  
 394 001724 000167 177104 100\$:  
 395  
 396  
 397  
 398  
 399  
 400  
 401 001730  
 402 001739 012705 001752:  
 403 001734 012737 006412 000000G  
 404 001742 004767 096222  
 405 001746 000167 177062:  
 406  
 407 001752 000002 000000G 000000G  
 001760 000000G

;THIS PORTION WILL READ THF "REAL" PARAMETERS, SUCH AS  
;SIGNAL AND SAMPLE FREQUENCY; THIS IS NO LONGER USED.  
;DATA INPUT IS FOUR BYTES WHICH ARE MORE LITERALLY A  
;32-BIT FLOATING VARIABLE IN PDP-11 FORMAT.

FREQP:  
 CJP ;FS,0\*1C ;IS THE CODE "FS"?  
 BNE 1,\$  
 MOV C\*IP,0\*FSIG ;SIGNAL FREQUENCY  
 MOV C\*IP+2,0\*FSIG+2  
 BR 100\$

FCP,0\*1C ;IS THE CODE "FC"?  
 20\$  
 MOV C\*IP,0\*FSAM ;SAMPLE FREQUENCY  
 MOV C\*IP+2,0\*FSAM+2  
 BR 100\$

IMP GEAR ;UNPACKAGED; ERROR!  
 JMP RDCODE ;RETURN FOR MORE INPUT

;ROUTINE FOR HANDLING "MESSAGE OF THE DAY"  
;HOST MAY SEND UP TO 80-CHARACTER MESSAGE TO SLAVF TO  
;BE DISPLAYED ON THE VOLTMETER

MESDAY:  
 MOV ,0\*MESARG,R5 ;REPLACE "00" WITH CR-LF  
 MOV ,6412,0\*1C ;OPTIONAL STRING;  
 JSR PC,WHITE  
 RDCODE.

```

409
410
411
412 001762 013760 006000C          !THIS SECTION WILL DECODE THE BYTE STRING FOLLOWING A
413 001766 162700 0090002          ;SETUP COMMAND INTO AN INTEGER STORED IN R3.
414 001772 012791 0090003          !LENGTH OF RECEIVED STRING
415 001776 005C93 0000000          !LESS CODE
416 001776 005C93 0000000          !ADDRESS OF RECEIVED STRING
417 002000 0050002          !CLEAR ACCUMULATED DATA WORD
418 002000 112102 0000000          !TO DE-SAFE
419 002002 162702 0000000          !GET AN ASCII BYTE
420 002004 002010 0000000          !MAKE IT A BINARY DIGIT
421 002010 002003 0000000          !AND ADD TO DATA WORD
422 002012 005300 0000000          !DONE YET?
423 002014 003403 0000000          !FINISHED
424 002016 006327 0000012          !SHIFT LEFT
425 002022 000766 0000000          !NEXT DIGIT
426 002024 000207 0000000          !PC
427
428
429
430
431
432
433 002026 010046          !RUDIMENTARY HANDLING OF FLOATING-POINT EXCEPTIONS
434 002030 010146          ;1. E. HALT ON ERROR: CLEAN UP ON CONTINUE (PROCEED)
435 002032 170200          ;FLOATING-POINT INTERRUPT SERVICE
436 002034 170300          !STORE KLT-11A FPS
437 002036 006900          !STORE .E1-11A FEC
438 002040 100000          !CLEAR ERROR BIT
439 002044 042700 100000          !R0,-(SP)
440 002044 170100          !R1,-(SP)
441 002046 012601          !R0
442 002050 012600          !R1
443 002052 006002          !RT1
444 002054 000000          !RT1
445 002054 000000          ;ERROR INTERRUPTS FROM (FORTRAN) FUNCTIONS
446 002056 000002          ;EG SQRAN2, SSIN, SSQRT

```

448  
449  
450  
451 002060 023727 000000G 051522 EXEC: CMP @\*IC,.\*RS  
452 002060 001002 000000G 0001022 BNE 108 ;IS THE CODE 'RS'?  
453 002066 001002 000000G 0001022 JMP WAKE ;RESTART THE CONVERSION COUNT  
454 002070 000167 000562  
455 002074 023727 000000G 042515 108: CMP @\*IC,.\*ME  
456 002074 001002 000000G 042515 108: BNE 208 ;IS THE CODE 'ME'?  
457 002102 001002 000000G 0001022 JMP MEASURE  
458 002104 000167 000402  
459 002110 023727 000000G 044514 208: CMP @\*IC,.\*LI  
460 002116 001002 000000G 044514 208: BNE 308 ;IS THE CODE 'LI'?  
461 002116 000167 000552 JMP PARSLST ;DO A/D CONVERSION  
462 002120 000167 000764  
463 002124 023727 000000G 050122 308: CMP @\*IC,.\*RP  
464 002132 001002 000000G 046526 408: BNE 408 ;LIST PARAMETERS ON SLAVE  
465 002134 000167 000764 JMP PARSND ;SEND PARAMETERS TO HOST  
466 002140 023727 000000G 046526 408: CMP @\*IC,.\*VM  
469 002146 001002 000000G 0001022 508: BNE 508 ;IS THE CODE 'VM'?  
470 002150 000167 003750 JMP VNDISP ;START VOLTMETER MODE  
471 002154 123727 000000G 0001122 508: CMP @\*IC,.\*R  
472 002162 001002 000000G 0001122 508: BNE 708 ;IS THE CODE 'R'??  
473 002162 000167 000760 JMP ABSEND ;SEND WAVEFORM TO HOST  
474 002164 000167 000760  
475 002170 023727 000000G 044527 708: CMP @\*IC,.\*WI  
476 002176 001002 000000G 044527 708: BNE 808 ;IS THE CODE 'WI'?  
477 002176 000167 002072 JMP WINDIN ;INPUT A WINDOW  
478 002200 000167 000760  
479 002204 023727 000000G 044524 808: CMP @\*IC,.\*TI  
480 002212 001002 000000G 044524 808: BNE 908 ;IS THE CODE 'TI'?  
481 002214 000167 176614 JMP RDCODE ;SEND  
482 002214 000167 000760  
483 002220 023727 000000G 053503 908: CMP @\*IC,.\*CW  
484 002220 001002 000000G 053503 908: BNE 1008 ;IS THE CODE 'CW'?  
485 002230 000167 001460 JMP CHAVE ;COMPUTATION MODES?  
486 002234 123727 000000G 000102 1008: CMP @\*IC,.\*B  
487 002242 001424 BEQ 1108 ;IS THE CODE 'B'??  
488 002244 123727 000000G 000103 CMP @\*IC,.\*C  
489 002252 001420 BEQ 1109 ;IS THE CODE 'C'? OR 'D'?  
490 002254 123727 000000G 000104 CMP @\*IC,.\*D  
491 002264 001414 BEQ 1109 ;IS THE CODE 'D'??  
492 002264 123727 000000G 000105 CMP @\*IC,.\*E  
493 002272 001410 BEQ 1109 ;IS THE CODE 'E'?  
494 002274 123727 000000G 000106 CMP @\*IC,.\*F  
495 002274 001404 BEQ 1109 ;IS THE CODE 'F'??  
496 002274 001404 BEQ 1109 ;IS THE CODE 'F'??  
497 002302 001404 BEQ 1109 ;IS THE CODE 'PO'?  
498 002304 023727 000000G 047520 CMP @\*IC,.\*PO  
499 002312 001002 BNE 1208 ;IS THE CODE 'PO'?  
500 002312 000167 001040 JMP CPUT ;COMPUTE AND RETURN DATA TO HOST  
501 002314 000167 000000 1208: JMP CERU ;CODE NOT FOUND: ERROR  
502 002320 000167 000000

505  
506  
507      0023224      012705      002360\*  
508      002330      004767      005334  
509      002330      004767      005334  
510      002334      002765      002350\*  
511      002340      004767      005324  
512  
513      002344      000167      176464  
514  
515      002359      003002      000090C 003000G CERIA:  
      002356      000000G  
516  
517      002360      003002      002747\* 002406\* CERIA:  
      002366      000000G  
518      002370      0112      0115      0103 CERIA:      .ASCII <12><15>"CODE ERROR:  
      002373      0117      0104      0105  
      002376      040      0105      0122  
      002401      0122      0117      0122  
      002404      072      0146  
519  
520      002406      000016 CERIA:      EVEN  
      521      000000G CERIA:      WORD 14.  
522  
523  
524      002410      012706      000000G  
525  
526      002414      012705      001112\*  
527  
528      002414      004767      000000C  
529      002410      012705      001112\*  
530      002410      012706      000000G  
531      002414      012705      001112\*  
532      002420      004767      000000C  
533      002424      012705      002446\*  
534      002430      004767      005534\*  
535      002434      012700      000000  
536      002440      106409      176366  
537      002442      000167  
538  
539      002446      000002      002456\* 002516\* ABO:  
      002454      000000G  
540      002456      0115      0152      0152 ABMES:      .ASCII <13>\*\*\* FUNCTION ABORTED \*\*\*"  
      002461      0152      040      006  
      002464      0125      016      003  
      002467      0124      011      017  
      002472      0116      0140      011  
      002475      0192      0117      012  
      002500      0124      0105      0194  
      002503      0149      0152      0152  
      002506      0152  
541  
542      002510      000031 ABLES:      EVEN  
      002510      0100      000000G  
      002510      0100      000000G

;THIS PORTION WILL TYPE OUT ERROR MESSAGES:  
;CONDITION? UNRECOGNIZED CODE.

CERIA:      MOV \*CERA,R5  
                JSR PC,WRITE  
                MOV \*CERC,R5  
                JSR PC,WRITE  
                JRP RDCode      ;PRINT ERROR MESSAGE  
                ;PRINT ERRONEOUS CODE  
                ;PRINT R2 FOR MORE INPUT

;THIS PORTION WILL HANDLE IRREGULAR INTERRUPTIONS.  
;IE, PROCESS ABORTS FROM THE HOST PROCESSOR.  
;ACTIVATED BY DEVICE TRIGGER ('GET') ON BUS.  
;ENTERED FROM CP10SR / INTERRUPT SERVICE ROUTINE

ABORT:      MOV \*STACK,SP  
                MOV \*GP1NA,R5  
                JSR PC,GPINI  
                MOV \*ABO,R5  
                JSR PC,WRITE  
                MOV \*O,RO  
                HTPS  
                JRP RDCode      ;INITIALIZE STACK  
                ;INITIALIZE IEEE-488 BUS CARD  
                ;PRINT MESSAGE  
                ;CLEAR PSW  
                ;ENABLE INTERRUPTS  
                ;RETURN FOR INPUT

2,ABMES,ABLEN,DLUN

ABMES:      .WORD 26.

```

544
545
546
547
548
549
550
551 002512    023737  30112  ' 006000G   :THIS PORTION OF THE PROGRAM CONTROLS THE A/D CONVERSION
552 002512    023737  30112  ' 006000G   ;AND DMA LOADING OF MEMORY WITH DATA.
553 002520    904064    006000G   :CHECK POINTS REQUESTED AGAINST BUFFER SIZE.
554 002522    112737  177677  000000G   ;TRY A/D CONVERSION AND REPEAT IF ERROR OCCURS.
555 002530    000462    000000G   :INCREMENT CONVERSION (AND "IGNORE") COUNTER.
556
557 002532    004767  000022    10E.   :MEASURE:
558
559 002536    012705  002552.   :ACKNOWLEDGE THIS
560 002542    004767  000000G   :NO-PARAMETER ERROR (-101)
561 002546    000167  176262    :ACKNOWLEDGE THIS
562 002552    000002    000000G  010164'  :ACKNOWLEDGE THIS
563
564 002560    013737  000000C  000000G  DOAD:  :DO A/D CONVERSION
565 002566    012705  002636'   :ALLOW HOW MANY TRIES?
566 002572    004767  000000G   :AERRS, &ECNT
567 002576    005237  000000G   :ARCON, R5
568 002602    005737  000000H   :PC, A32D
569 002606    002403    000000G   :&ICA
570 002610    004767  004122    :&ISW
571 002614    0002097   308:   :HLT
572 002616    005237  000000G   :PC, CH0DIS
573 002622    004767  004110    :INC
574 002626    005337  000000G   :JSR
575 002632    003355    DEC
576 002634    000207    BCT
577
578 002636    000006    000000G  000000G  ADCOM:  :RTS
579 002644    000000G  000000G  000000G   6. ADBUF, GAIN, PAD, ADS, GATED, ADIC, ISW
580
581
582
583
584 002656    005037  000000G   :THIS PORTION OF THE PROGRAM INITIALIZES THE TWO MAIN
585 002656    005037  000000G   ;COUNTERS TO ZERO AND PRINTS THE "WAKE UP" TEXT"
586 002662    005037  000000G   :CLEAR A/D CONVERSION COUNTER
587 002666    004767  004044    :CLEAR A/D ERROR COUNTER
588 002672    000167  176136    :DISPLAY TEXT
589

```

591  
592  
593 002676 012737 000025 000000C  
594 002676 012737 000001 000000C  
595 002676 012737 0000002 000000C  
596 002704 012737 0000001 000000C  
597 002712 012737 000000G 000000G  
598 002720 012705 003042 005240  
599 002724 004767 005240  
600 002730 012705 003022  
601 002734 004767 005320  
602 002740 012795 003064  
603 002744 004767 005220  
604 002750 012705 003032  
605 002754 017737 000000G 000000G  
606 002762 004767 005222  
607 002766 005237 000000C  
608 002772 005237 000000C  
609 002776 005237 000000C  
610 003002 005337 000000G  
611 003006 003493  
612 003010 004767 000066  
613 003014 000741  
614  
615 003016 000167 176012  
616  
617 003022 000002 000000G 0000002  
618 003030 000000G  
619 003032 000002 000000G 0000006  
620 003040 000000G  
621 003042 000002 003052 003062  
622 003050 000000C  
623 003052 015 012 104  
624 003055 101 124 101  
625 003060 040 050  
626  
627 003100 000004  
628  
629 003102 012700 100000  
630 003102 012700 100000  
631 003106 005300  
632 003110 000240  
633 003112 000240  
634 003114 000240  
635 003116 000240  
636 003120 001372  
637 003122 000297

;THIS PORTION OF THE PROGRAM WILL LIST THE MAIN SETUP  
;PARAMETERS ON THE LSIVM CONSOLE.

PARLIST:  
 MOV \*21,0,LCTN  
 MOV \*1,0,LPTH  
 ;NUMBER TO PRINT  
 ;STARTING AT PARAMETER 1  
 ;AT ADDRESS "DATA".  
 MOV \*DATA,0,LADD  
 MOV \*LSTN,0,R5  
 JSR PC,WRITE  
 ;PRINT INITIAL DESCRIPTION  
 MOV \*LSTN,0,R5  
 JSR PC,PRINT  
 ;PRINT VARIABLE INDEX  
 MOV \*LST2,0,R5  
 JSR PC,WRITE  
 ;PRINT INTERNAL PUNCTUATION  
 MOV \*LSTD,0,R5  
 JSR PC,PRINT  
 ;PRINT INTEGER VARIABLE  
 INC \*LADD  
 INC \*LADD  
 INC \*LPTR  
 DEC \*LCTR  
 ;ANOTHER ONE?  
 JSR PC,IFWAIT  
 BR 10\$  
 ;YES, WAIT TO SEE THIS ONE.  
 10\$:  
 ;RETURN FOR MORE INPUT

JMP RDCODE  
 LSTN0:  
 2,LPTR,2,DLUN  
 LSTDA:  
 2,LBAT,6,DLUN  
 LSTI:  
 2,LST1,LST1,DLUN  
 LIST1:  
 .ASCII <15><12>"DATA ("  
 EVEN  
 LSTL1:  
 .WORD LSTL1-LIST1  
 LST2:  
 2,LST2,LSTL2,DLUN  
 LIST2:  
 .ASCII " ) = "  
 EVEN  
 LSTL2:  
 .WORD LSTL2-LIST2  
 IFWAIT:  
 MOV \*100000,R0  
 R0  
 ;DISPLAY ROUTINE (1 SEC??)  
 NOP  
 NOP  
 NOP  
 NOP  
 BNE 10\$  
 RTS  
 PC

639  
640  
641  
642 003124 012703 003140' ;THIS PORTION OF THE PROGRAM IS DESIGNED TO SEND THE DATA  
643 003124 004767 0000000C ;PARAMETERS BACK TO THE HOST PROCESSOR FOR EXAMINATION.

644 003130 004767 0000000C	PARSND:	MOV R5 JSR PC, GPOUT	:OUTPUT DATA
645 003134 009167 175674 PARA: RDCODE	JMP 2, DATA, PSIZE	:RETURN FOR MORE INPUT	
646 003134 009167 175674 PARA: WORD, 42.			
647 003140 0000002 000000G 003146' ;THIS PORTION OF THE PROGRAM IS DESIGNED TO SEND ONE OF THE 648 003146 0000052 ;THREE DATA BUFFERS BACK TO THE HOST PROCESSOR. THE INPUT ;CODES ARE R1 (BUFFER 1), R2 (BUFFER 2), R3 (BUFFER 3). ;THE BUFFERS CONTAIN 12-BIT BINARY WORDS.			
649			
650			
651			
652			
653			
654			
655			
656			
657 003150 013737 003356' ;ALSO TO TRANSFER BACK THE WINDOW (RW)--FLOATING WORDS. 658 003150 023727 0000004C ;PRECEDED BY COMPUTED AVERAGE.	ADSEND:	MOV @RAA, @RAA+4 CMP @IC, #RW BNF 5\$	:LOAD ADDRESS INTO RAM ;WRITE WINDOW? ;NO
659 003156 023727 0000000C			
660 003164 001015			
661 003166 013702 0000000C			
662 003172 005202			
663 003174 006302			
664 003176 006302			
665 003200 010237 0000000C			
666 003204 012705 003350'			
667 003210 004767 0000000C			
668 003214 000167 175614			
669			
670 003220 012701 0000000C	5\$:	MOV @DBUF, R1 CMP @IC, #R1 BEQ 30\$	:ABUFFER, R1 ;WHAT CODE? (BUFFER?)
671 003224 023727 0000000C 030522			
672 003232 001422			
673 003234 023727 0000000G 031122			
674 003242 001412			
675 003244 023727 0000000C 031522			
676 003252 001402			
677 003254 000167 177644			
678 003260 063701 0000000C	10\$:	ADD @PAD, R1 ADD @PAD, R1 ADD @PAD, R1	:ADD IN OFFSETS FOR ;DOUBLE FOR BYTES
679 003264 063701 0000000C			
680 003270 063701 0000000C	20\$:	ADD @PAD, R1 ADD @STPT, R1 ADD @STPT, R1	:...FOR BUFFERS 2 AND 3 ;AND FOR THE STARTING POINT ;TWICE--WORD ADDRESS!
681 003274 063701 0000000G			
682 003300 063701 0000000C	30\$:	MOV R1, @RAA+2	
683 003304 063701 0000000C			
684 003310 010137 0000002C			
685			
686 003314 013702 0000000C			
687 003320 060202			
688 003322 010237 0000000G			
689 003326 0127037 0000000C			
690 003334 012705 0000000C			
691 003340 004767 0000000C			
692			
693 003344 009167 175464	JMP 2, WAV, ADWL		
694 003350 009002 0000000C ADWA:	RAN:		
695 003356 0000000C			

```

647          COMPUTATIONAL EXECUTIVE
648          1. CONVERT DATA TO REAL, SCALING NUMBERS
649          2. MULTIPLY SEQUENCE BY WINDOW IF DESIRED
650          3. AVERAGE SEQUENCE TO FEWER SAMPLES IF DESIRED
651          4. REQUEST SPECIFIC CONFIGURATION TYPE
652          5. DATA IS SCALED FOR GAIN IN A/D CONVERTER
653          6. DATA IS CORRECTED FOR WINDOW ATTENUATION IF NECESSARY.
654          7. TRANSMIT COMPUTED DATA TO HOST.

715          COMPUTATIONAL EXECUTIVE
707 003369 923737 001120' 0000000C      CR*
698          1. CONVERT DATA TO REAL, SCALING NUMBERS
699          2. MULTIPLY SEQUENCE BY WINDOW IF DESIRED
700          3. AVERAGE SEQUENCE TO FEWER SAMPLES IF DESIRED
701          4. REQUEST SPECIFIC CONFIGURATION TYPE
702          5. DATA IS SCALED FOR GAIN IN A/D CONVERTER
703          6. DATA IS CORRECTED FOR WINDOW ATTENUATION IF NECESSARY.
704          7. TRANSMIT COMPUTED DATA TO HOST.

716 003366 923737 001120' 0000000C      CR*
708 003366 923737 001120' 0000000C      CR*
709 003379 000000 0000000C      CR*
710 003372 000117 175436      CR*
711          COMPUTATIONAL EXECUTIVE
712 003376 113700 0000000C      CR*
713 003402 042700 177774      CR*
714 003406 003300      CR*
715 003410 006190      CR*
716 003412 006190      CR*
717 003414 010037 0000000C      CR*
718          COMPUTATIONAL EXECUTIVE
719 003420 013797 0000000C 0000000C      CR*
720 003426 013737 0000000C 0000000C      CR*
721 003434 023727 0000000C 047520      CR*
722 003442 001510      CR*
723 003444 023727 0000000C 045193      CR*
724 003452 001512      CR*
725 003454 004767 000442      CR*
726 003460 123727 0000000C 000106      CR*
727 003466 001470      CR*
728 003470 005737 0000000C      CR*
729 003474 001492      CR*
730 003476 004767 000654      CR*
731 003502 005737 0000000C      CR*
732 003506 001462      CR*
733 003510 004767 000704      CR*
734          COMPUTATIONAL EXECUTIVE
735 003514 123727 0000000C 000102      CR*
736 003522 001422      CR*
737 003524 123727 0000000C 000103      CR*
738 003532 001424      CR*
739 003534 123727 0000000C 000104      CR*
740 003542 001426      CR*
741 003544 123727 0000000C 000105      CR*
742 003552 001430      CR*
743 003554 123727 0000000C 000106      CR*
744 003562 001432      CR*
745 003564 000167 176534      CR*

647          COMPUTATIONAL EXECUTIVE
648          1. CONVERT DATA TO REAL, SCALING NUMBERS
649          2. MULTIPLY SEQUENCE BY WINDOW IF DESIRED
650          3. AVERAGE SEQUENCE TO FEWER SAMPLES IF DESIRED
651          4. REQUEST SPECIFIC CONFIGURATION TYPE
652          5. DATA IS SCALED FOR GAIN IN A/D CONVERTER
653          6. DATA IS CORRECTED FOR WINDOW ATTENUATION IF NECESSARY.
654          7. TRANSMIT COMPUTED DATA TO HOST.

715          COMPUTATIONAL EXECUTIVE
707 003369 923737 001120' 0000000C      CR*
698          1. CONVERT DATA TO REAL, SCALING NUMBERS
699          2. MULTIPLY SEQUENCE BY WINDOW IF DESIRED
700          3. AVERAGE SEQUENCE TO FEWER SAMPLES IF DESIRED
701          4. REQUEST SPECIFIC CONFIGURATION TYPE
702          5. DATA IS SCALED FOR GAIN IN A/D CONVERTER
703          6. DATA IS CORRECTED FOR WINDOW ATTENUATION IF NECESSARY.
704          7. TRANSMIT COMPUTED DATA TO HOST.

716 003366 923737 001120' 0000000C      CR*
708 003366 923737 001120' 0000000C      CR*
709 003379 000000 0000000C      CR*
710 003372 000117 175436      CR*
711          COMPUTATIONAL EXECUTIVE
712 003376 113700 0000000C      CR*
713 003402 042700 177774      CR*
714 003406 003300      CR*
715 003410 006190      CR*
716 003412 006190      CR*
717 003414 010037 0000000C      CR*
718          COMPUTATIONAL EXECUTIVE
719 003420 013797 0000000C 0000000C      CR*
720 003426 013737 0000000C 0000000C      CR*
721 003434 023727 0000000C 047520      CR*
722 003442 001510      CR*
723 003444 023727 0000000C 045193      CR*
724 003452 001512      CR*
725 003454 004767 000442      CR*
726 003460 123727 0000000C 000106      CR*
727 003466 001470      CR*
728 003470 005737 0000000C      CR*
729 003474 001492      CR*
730 003476 004767 000654      CR*
731 003502 005737 0000000C      CR*
732 003506 001462      CR*
733 003510 004767 000704      CR*
734          COMPUTATIONAL EXECUTIVE
735 003514 123727 0000000C 000102      CR*
736 003522 001422      CR*
737 003524 123727 0000000C 000103      CR*
738 003532 001424      CR*
739 003534 123727 0000000C 000104      CR*
740 003542 001426      CR*
741 003544 123727 0000000C 000105      CR*
742 003552 001430      CR*
743 003554 123727 0000000C 000106      CR*
744 003562 001432      CR*
745 003564 000167 176534      CR*

647          COMPUTATIONAL EXECUTIVE
648          1. CONVERT DATA TO REAL, SCALING NUMBERS
649          2. MULTIPLY SEQUENCE BY WINDOW IF DESIRED
650          3. AVERAGE SEQUENCE TO FEWER SAMPLES IF DESIRED
651          4. REQUEST SPECIFIC CONFIGURATION TYPE
652          5. DATA IS SCALED FOR GAIN IN A/D CONVERTER
653          6. DATA IS CORRECTED FOR WINDOW ATTENUATION IF NECESSARY.
654          7. TRANSMIT COMPUTED DATA TO HOST.

715          COMPUTATIONAL EXECUTIVE
707 003369 923737 001120' 0000000C      CR*
698          1. CONVERT DATA TO REAL, SCALING NUMBERS
699          2. MULTIPLY SEQUENCE BY WINDOW IF DESIRED
700          3. AVERAGE SEQUENCE TO FEWER SAMPLES IF DESIRED
701          4. REQUEST SPECIFIC CONFIGURATION TYPE
702          5. DATA IS SCALED FOR GAIN IN A/D CONVERTER
703          6. DATA IS CORRECTED FOR WINDOW ATTENUATION IF NECESSARY.
704          7. TRANSMIT COMPUTED DATA TO HOST.

716 003366 923737 001120' 0000000C      CR*
708 003366 923737 001120' 0000000C      CR*
709 003379 000000 0000000C      CR*
710 003372 000117 175436      CR*
711          COMPUTATIONAL EXECUTIVE
712 003376 113700 0000000C      CR*
713 003402 042700 177774      CR*
714 003406 003300      CR*
715 003410 006190      CR*
716 003412 006190      CR*
717 003414 010037 0000000C      CR*
718          COMPUTATIONAL EXECUTIVE
719 003420 013797 0000000C 0000000C      CR*
720 003426 013737 0000000C 0000000C      CR*
721 003434 023727 0000000C 047520      CR*
722 003442 001510      CR*
723 003444 023727 0000000C 045193      CR*
724 003452 001512      CR*
725 003454 004767 000442      CR*
726 003460 123727 0000000C 000106      CR*
727 003466 001470      CR*
728 003470 005737 0000000C      CR*
729 003474 001492      CR*
730 003476 004767 000654      CR*
731 003502 005737 0000000C      CR*
732 003506 001462      CR*
733 003510 004767 000704      CR*
734          COMPUTATIONAL EXECUTIVE
735 003514 123727 0000000C 000102      CR*
736 003522 001422      CR*
737 003524 123727 0000000C 000103      CR*
738 003532 001424      CR*
739 003534 123727 0000000C 000104      CR*
740 003542 001426      CR*
741 003544 123727 0000000C 000105      CR*
742 003552 001430      CR*
743 003554 123727 0000000C 000106      CR*
744 003562 001432      CR*
745 003564 000167 176534      CR*

```

```

747 :COMPUTE AS SPECIFIED.....
748
749 :RETURN DATA SPECIFIED.....
750
751 003579 004767 001146 :COMPUTE AMPLITUDE, PHASE OF HARMONIC
    JSR PC,CSL
    JSR PC,TDF
    JMP RDGDE
    ;RETURN DATA TO HOST

752 003600 006167 175236
753
754 003634 004767 001054 :COMPUTE AMPLITUDE, PHASE OF FUNDAMENTAL
    JSR PC,GDFT
    JSR PC,TDF
    JMP RDGDE
    ;RETURN DATA TO HOST

755 003610 004767 001242
756 003614 006167 175214
757
758 003620 004767 000764 :COMPUTE AMPLITUDE, PHASE, DISTORTION
    JSR PC,GDFTD
    JSR PC,TDFTD
    JMP RDGDE
    ;RETURN DATA TO HOST

759 003624 004767 001204
760 003636 006167 175200
761
762 003634 004767 001342 :COMPUTE TIME-AVERAGED PARAMETERS
    JSR PC,GRISV
    JSR PC,TRNSV
    JMP RDGDE
    ;RETURN DATA TO HOST

763 003640 004767 001514
764 003644 006167 175164
765
766 003650 004767 001420 :COMPUTE ABSOLUTE PEAK DATA
    JSR PC,GPKEKO
    JSR PC,TPEK
    JMP RDGDE
    ;RETURN DATA TO HOST

767 003654 004767 001544
768 003660 006167 175150
769
770 003664 004767 001644 :GET POWER IN CHANNELS #1-#2
    JSR PC,GPWR
    JSR PC,TPWR
    JMP RDGDE
    ;RETURN DATA TO HOST

771 003670 004767 002920
772 003674 006167 175134
773
774 003700 004767 002639 :GET JOINT MOMENT AND COVARIANCE
    JSR PC,CJC
    JSR PC,TJC
    JMP RDGDE
    ;RETURN DATA TO THE HOST

775 003704 004767 002174
776 003710 006167 175120

```

```

778          7B3 003714    012769  000000G      Cwave:      ;START OF FIRST BUFFER
779          7B4 003720    063709  000000C      MOV     R0      ;TRUE START OF FIRST WAVEFORM
780          7B5 003724    063700  000000C      ADD     R0      ;ADD STARTING POINT--TWICE FOR BYTES)
781          7B6 003730    063700  000000C      ADD     R0      ;ADD PTR, R0
782          7B7 003734    063700  000000C      ADD     R0      ;ADD PTR, R0
783          7B8 003740    013761  000000G      MOV     R0      ;(ALSO TWICE),
784          7B9 003744    063701  000000C      ADD     R0      ;FIND END OF FIRST WAVEFORM
785          790 003750    060001  000000I      ADD     R1      ;BY ADDING TWICE POINTS TO A/D
786          791 003752    013702  000000G      ADD     R0      ;TRUE END OF SECOND WAVEFORM
787          792 003756    063702  000000C      ADD     R0      ;FIND END OF LAST WAVEFORM
788          793 003762    069102      ADD     R1      ;BY ADDING TWICE POINTS TO A/D
789          794          170001      ADD     R1      ;TRUE END OF THIRD WAVEFORM

795          796 003766    177237  000000G      SETF   AC2      ;NCYC FLOATED
796          797 003772    177137  000000C      LDC1F AC1      ;PTR FLOATED
797          798 003776    174601  000000I      DIVF   AC1, AC2      ;NCYC/PTR
798          799 004000    171237  004112*      MULF   AC1, AC2      ;2*PI*(NCYC/PTR)

800          B01 004004    013793  000000G      108:      POINTS TO COMPUTE
801          B02 004010    177003  000000G      NOV    R3, AC0      ;FLOAT I
802          B03 004012    172067  000100      ADDF  N1, AC0      ;(1-1)
803          B04 004016    171902  010046      MULF  AC2, AC0      ;(1-1)*(--)
804          B05 004020    010046      NOV    R0, -(SP)      ;SAVE REGISTERS ON STACK
805          B06 004022    010146      NOV    R1, -(SP)
806          B07 004024    010246      NOV    R2, -(SP)
807          B08 004026    010346      NOV    R3, -(SP)
808          B09 004030    174246      STF   AC2, -(SP)
809          B10 004032    004767  000000G      JSR   PC,$$COS      ;COS((2*PI*(1-1)*NCYC/PTR)
810          B11 004036    174437  004272      DIVF  Q*FFAC, AC0      ;(1-1)*2048./10.
811          B12 004042    172626      LDIF  (SP)+, AC2      ;RESTORE REGISTERS FROM STACK
812          B13 004044    012603      MOVF  (SP)+, R3
813          B14 004046    012602      MOVF  (SP)+, R2
814          B15 004050    012601      MOVF  (SP)+, R1
815          B16 004052    012600      MOVF  (SP)+, R0

816          B17          171137  000000G      LDC1F AC1      ;FLOAT GAIN (SCALE)
817          B18          177137  000000G      MULF AC0, AC1      ;GAIN(1)*(--)
818          B19 004054    177137  000000G      STCF1 AC1, -(R0)      ;SIMULATED A/D CONVERTED VALUE
819          B20 004060    171109      LDC1F AC1      ;AGAIN
820          B21 004062    175540      MULF AC0, AC1
821          B22          177137  000000C      STCF1 AC1, -(R1)
822          B23 004064    177137  000000C      LDC1F AC1      ;AGAIN
823          B24 004070    171160      MULF AC0, AC1
824          B25 004072    175541      STCF1 AC1, -(R2)
825          B26          177137  000000G      SOB   R3, 108      ;NEXT SAMPLE (WORKING FROM END TO FRONT)
826          B27 004074    177137  000000C      JMP   RDODE
827          B28 004100    171100      FLT2  6, 2631054
828          B29 004102    175542      H1:   -1.
829          B30          146260  0000000

```

ROUTINE TO "FLOAT" THE A/D DATA BUFFER FOR PROCESSING  
 ; CONVERT INTEGER TO FLOATING NUMBERS IN VOLTS IN RBUF).

```

836          F1.C1:      MOV    R0,-(SP)      ;SAVE REGISTERS ON STACK
837          F1.C1:      MOV    R1,-(SP)
838          F1.C1:      MOV    R2,-(SP)
839 004122   010046      SETF
840 004122   010046      MOV    #ADBUF,R1
841 004124   010046      CMPB  #01CN,'1
842 004126   010046      BEQ   30$      ;STARTING WHERE?
843 004130   170001      CMPB  #01CN,'2
844          F1.C1:      BEQ   20$      ;STORE ADDRESS
845 004132   012701      000000C 000000C 000000C 000000C 000000C 000000C
846 004136   123727      000000C 000000C 000000C 000000C 000000C 000000C
847 004144   001422      000000C 000000C 000000C 000000C 000000C 000000C
848 004146   123727      000000C 000000C 000000C 000000C 000000C 000000C
849 004154   001412      000000C 000000C 000000C 000000C 000000C 000000C
850 004156   123727      000000C 000000C 000000C 000000C 000000C 000000C
851 004164   001462      000000C 000000C 000000C 000000C 000000C 000000C
852 004166   000167      176132      000000C 000000C 000000C 000000C 000000C 000000C
853 004172   063701      000000C 000000C 000000C 000000C 000000C 000000C
854 004176   063701      000000C 000000C 000000C 000000C 000000C 000000C
855 004202   063701      000000C 000000C 000000C 000000C 000000C 000000C
856 004206   063701      000000C 000000C 000000C 000000C 000000C 000000C
857 004212   063701      000000C 000000C 000000C 000000C 000000C 000000C
858 004216   063701      000000C 000000C 000000C 000000C 000000C 000000C
859          F1.C1:      ADDI  R1,10$      ;ADD IN OFFSETS FOR
860 004222   013700      000000C 000000C 000000C 000000C 000000C 000000C
861 004226   012702      000000C 000000C 000000C 000000C 000000C 000000C
862 004232   042711      170000      000000C 000000C 000000C 000000C 000000C 000000C
863 004236   021127      004000      000000C 000000C 000000C 000000C 000000C 000000C
864 004242   002402      000000C 000000C 000000C 000000C 000000C 000000C
865 004244   052711      170000      000000C 000000C 000000C 000000C 000000C 000000C
866          F1.C1:      ADDI  R1,20$      ;FOR BUFFERS 2 AND 3
867 004250   177021      000000C 000000C 000000C 000000C 000000C 000000C
868 004252   171037      004272      000000C 000000C 000000C 000000C 000000C 000000C
869          F1.C1:      ADDI  R1,30$      ;AND FOR THE STARTING POINT
870 004256   174022      000000C 000000C 000000C 000000C 000000C 000000C
871 004260   077014      000000C 000000C 000000C 000000C 000000C 000000C
872          F1.C1:      LDCLF(R1)+,AC9      ;CLEAR TO TWELVE BITS
873 004262   012602      000000C 000000C 000000C 000000C 000000C 000000C
874 004264   012601      000000C 000000C 000000C 000000C 000000C 000000C
875 004266   012600      000000C 000000C 000000C 000000C 000000C 000000C
876 004270   000207      000000C 000000C 000000C 000000C 000000C 000000C
877          F1.C1:      FFAC: .FLT2 .0048828125      i = 10. / 2048.

```

11SIIVTI MACRO M1112 23-JUL-82 14:13 PAGE 19

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

```

988      ;COMPUTE THE VOLTAGES VIA DISCRETE FOURIER TRANSFORMS (DFT):
989      ;TYPE "B"--VOLTAGE AND PHASE OF HGTU HARMONIC;
990      ;TYPE "C"--VOLTAGE AND PHASE OF FUNDAMENTAL;
991      ;TYPE "D"--VOLTAGE, PHASE, AND DISTORTION OF FUNDAMENTAL.
992      ;VOLTAGES ARE CORRECTED FOR A/D GAIN AND ARE RMS.
993      ;DISTORTION IS IN PERCENTAGE.

994      GNDT:    994610      ;ANAG,R5      ;COMPUTE THE DFT VALUES AND DISTORTION
995      994610      012705      994642'      ;PC,DAPD      ;DGRTZL-TYPE
996      994614      004767      0000000C      ;@*VDFT,AC0      ;CORRECT FOR A/D GAIN
997      994614      004767      0000000C      LDF
998      994620      172437      0000000C      JSR,DEGAIN
999      994624      004767      0000000C      PC,DEWIND
1000     994630      004767      0000000C      JSR
1001     994634      174037      0000000C      STF,AC0,@*VBFT
1002     994640      00000007      0000000C      RTS
1003     994642      00000007      0000000C      PC
1004     994650      0000000C      0000000C      7.RBUF,PTC,NCYF,VDFT,PBFT,DIST,HAR,IERD
1005     994656      0000000C      0000000C      AMAG:
1006     994664      012705      004716'      0000000C
1007     994664      004767      0000000C      @ADFT,RS
1008     994670      004767      0000000C      PC,DAPD
1009     994674      172437      0000000C      @*VDFT,AC0
1010     994700      004767      0000000C      PC,DEGAIN
1011     994704      004767      0000000C      PC,DEWIND
1012     994714      00000007      0000000C      AC0,@*VBFT
1013     994716      00000007      0000000C      RTS
1014     994724      0000000C      0000000C      7.RBUF,PTC,NCYF,VDFT,PBFT,DIST,HAR,IERD
1015     994732      0000000C      004740'      0000000C
1016     994740      00000009      NONE:      .WORD 0
1017     994742      013705      0000000C      CSL:
1018     994742      004742      0000000C      @*HTC,R5
1019     994746      005205      0000000C      R5
1020     994750      070537      0000000C      @*NCYF,R5
1021     994754      010537      0000000C      R5,@*NCYHA
1022     994760      012705      995012'      ASL,R5
1023     994760      004764      004767      ;COMPUTE SFBFT
1024     994764      004767      0000000C      ;CORRECT FOR GAIN
1025     994770      172437      0000000C
1026     994774      004767      000470'      PC,DEGAIN
1027     995000      004767      000512'      PC,DEWIND
1028     995004      174037      0000000C      AC0,@*VBFT
1029     995010      00000007      RTS
1030     995012      00000007      0000000C      PC
1031     995012      00000007      0000000C      ASL:
1032     995020      0000000C      0000000C      R5
1033     995026      0000000C      004740'      ;AND FOR WINDOW IF USED
1034     995026      0000000C      PC
1035     995026      0000000C      7.RBUF,PTC,NCYHA,VDFT,PBFT,DIST,HAR,IERD

```

```

1033          ;ROUTINE TO RETURN DATA TO THE HOST.
1034          ;ALSO TO SET UP ON-LINE DISPLAY VALUES.
1035
1036  005034    TDFTD:      MOV    *TDFTA,R5
1037  005034    012705    005046'   ;OUTPUT VOLTAGE, PHASE, DISTORTION
1038  005040    004767    000000C
1039  005044    000415    000002   TDFTA:
1040  005046    000002    000000C  005054'   N12:
1041  005054    000014    .WORD 12.

1042
1043  005056    012705    005070'   TDFT:      MOV    *TDFTA,R5
1044  005056    004767    000000C   ;OUTPUT VOLTAGE, PHASE
1045  005062    000413    000002   TDFTA:
1046  005066    000002    000000C  005076'   NB:     2,DFD,NB
1047  005070    000010    .WORD 8.

1048  005076    000010    DSLD:      LDF    *DIST,AC@  ;LOAD DATA FOR DISPLAY ROUTINES
1049
1050  005100    172437    000000C  ;BASE ADDRESS FOR DISTORTIONS
1051  005104    012700    000000C  ;PARTICULAR ONE AVAILABLE
1052  005110    063700    000000C
1053  005114    174016    ADD    *CHNO,R@  ;LOAD IT
1054  005116    172437    000000C  ;LOAD IT
1055  005122    012700    000000C  ;SAME FOR VOLTAGE
1056  005126    063700    000000C
1057  005132    174010    ADD    *CHNO,R@  ;SAME FOR PHASE
1058  005134    172437    000000C
1059  005140    012700    000000C
1060  005144    063700    000000C
1061  005150    174010    ADD    *CHNO,R@  ;COMPUTE RELATIVE PHASE
1062  005152    172437    000000C
1063  005156    172537    000000C
1064  005162    173001    SUBF   *PHAS1,AC@  ;POSSIBLE--ONE CHANNEL DISPLAY
1065  005164    174037    000000C  AC1,AC@  ;POSSIBLE--TWO CHANNEL RELATIVE PHASE
1066
1067  005170    004767    001732   JSR    PC,CH1DIS
1068  005174    004767    002564   JSR    PC,CH3DIS
1069  005200    000207   RTS    PC

```

```

1071 :COMPUTE THE VOLTAGE PARAMETERS BY RMS TIME-AVERAGE:
1072 :TYPE 'E' -- GET RMS, PEAK, AND DC VOLTAGES.
1073 :TYPE 'F' -- GET ABS(PEAK) VOLTAGE ONLY.
1074 :VOLTAGES ARE CORRECTED FOR A/D CONVERTER GAIN.

1075
1076 005202 012795 005342'      GRMSV:      MOV    *ARM5V,R5      :CALL RMS COMPUTATION
1077 005202 004767 000000C      JSR    PC,CRMS
1078 005206 004767 000000C
1079 005212 172437 000000C      LDF    *RMSV,AC0      :CORRECT FOR A/D GAIN (RMSV)
1080 005216 004767 0000246      JSR    PC,DEGAIN
1081 005222 174037 000000C      STF    AC0,*RMSV
1082 005222 174037 000000C      LDF    *DCV,AC0      :CORRECT FOR A/D GAIN (DV)
1083 005226 172437 000000C      JSR    PC,DEGAIN
1084 005232 004767 0000232      STF    AC0,*DCV
1085 005236 174037 000000C      JSR    *PEKVP,AC0      :CORRECT FOR A/D GAIN (MAXIMUM)
1086 005232 172437 000000C      LDF    PC,DEGAIN
1087 005246 004767 0000216      JSR    AC0,*PEKVP
1088 005252 174037 000000C      STF    *PEKVM,AC0      :CORRECT FOR A/D GAIN (MINIMUM)
1089 005256 172437 000000C      LDF    PC,DEGAIN
1090 005262 004767 0000202      JSR    AC0,*PEKVM
1091 005266 174037 000000C      STF    PC
1092 005272 0000297      RTS

1093
1094 005274 012701 000000C      CPEKO:      MOV    *RBUF,R1      :FIND PEAK ONLY
1095 005274 012701 000000C      NOV   *PTC,R0
1096 005309 013709 000000C      LDF   (R1)+,AC0      :INITIALIZE PEAK
1097 005304 172421 000000C      ABSF  AC0
1098 005306 170600 000000C      DEC   3
1099 005310 005300 000000C      LD7   (R1)+,AC1      :GET ANOTHER VALUE
1100 005312 172521 000000C      ABSF  AC1
1101 005314 170601 000000C      CMPP  AC1,AC0      :ABSOLUTE VALUE
1102 005316 173401 000000C      CFCC
1103 005320 170600 000000C      BLE   208
1104 005322 003401 000000C      LDF   AC1,AC0      :NOPE, SAVE NEW PEAK
1105 005324 172401 000000C      SOB   R0,109      :YEP, NEXT VALUE
1106 005326 077007 000000C      JSR   PC,DEGAIN      :CORRECT FOR A/D GAIN
1107 005330 004767 000134 000000C      STF   AC0,*PEKVP
1108 005334 174037 000000C      RTS
1109 005340 0000267 000000C
1110
1111 005342 000006 000000C 000000C ARMSV:  6.PTC,RBUF,RMSV,DCV,PEKVP,PEKVM
1112 005350 000000C 000000C 000000C
1113 005356 000000C

```

```

1113          ;SUBROUTINES TO TRANSMIT RMS DATA TO THE HOST.
1114          ;ALSO PREPARATION FOR REAL-TIME DISPLAYS.
1115          ;  

1116  005360   012705   005414*    TRMSV:    MOV      *TRMSA,R5      ;TRANSFER THE DATA BACK
1117  005360   012705   004767      JSR      PC,GPOUT
1118  005364   004767   0000000C    LDF      @RHSV,AC0
1119  005370   012737   0000000C    MOV      @RHSV1,R0
1120  005374   012790   0000000C    ADD      @CHNO,R0
1121  005400   063760   0000000C    STF      AC0,(R0)
1122  005404   174010           PC,CH4DIS
1123  005406   004767   002154
1124  005412   000267           RTS      PC
1125  005414   000002   0000000G  TRMSA:  2,RMD,N16
1126  005422   000020           .WORD 16.
1127          ;  

1128  005424   003460*    TPEK:    MOV      *TPEKA,R5      ;TRANSFER PEAK VOLTAGE ONLY
1129  005424   012705   003460*    JSR      PC,GPOUT
1130  005430   004767   0000000C    LDF      @PEKVP,AC0
1131  005434   172437   0000000C    MOV      @RHSV1,R0
1132  005440   012760   0000000C    ADD      @CHNO,R0
1133  005444   063790   0000000C    STF      AC0,(R0)
1134  005450   174010           PC,CH4DIS
1135  005452   004767   002110
1136  005456   000267           RTS      PC
1137  005460   000002   0000000G  TPEKA:  2,PEKVP,R4
1138  005466   000004           .WORD 4.
1139          ;  

1140          ;SUBROUTINE TO CONVERT THE COMPUTED VOLTAGES TO TRUE VALUES
1141          ;BY REMOVING THE GAIN OF THE A/D CONVERTERS.
1142          ;  

1143          ;ALSO TO CORRECT DATA COMPUTED WITH A KAISER WINDOW, BY
1144          ;DIVIDING THE RESULTING MAGNITUDE BY THE WINDOW AVERAGE.
1145          ;  

1146          ;DEGAIN:    MOVB   @*1CN,R0      ;SUBROUTINE TO REMOVE CHANNEL GAIN
1147  005470   113700   0000000G    BIC      @17774,R0
1148  005470   042700   177774      ADD      R0,R0
1149  005474   042700   00000000    ADD      @GAIN-2,R0
1150  005500   066060   177766      SETF    LDClF  (R0),AC1
1151  005502   062260   177766
1152  005506   170061           DIVF    AC1,AC0
1153  005510   177110           RTS      PC
1154          ;  

1155  005512   174401           DEWIND:  TST      @*WODE
1156  005514   000207           BEQ      168
1157          ;SUBROUTINE TO REMOVE WINDOW FACTOR
1158  005516   005737   0000000C    LDF      @WAV,AC1
1159  005516   005737   0000000C    DIVF    AC1,AC0
1160  005522   001463           RTS      PC
1161  005524   172537   0000000G    LDF      @WAV,AC1
1162  005530   174401           DIVF    AC1,AC0
1163  005532   000207           RTS      PC
1164          ;  

1165          ;NOT THIS TIME
1166          ;GET AVERAGE (WAV)
1167          ;WDF = VDFT / WAV

```



```

1223
1224
1225
1226 005734 004767 177574      GJC:          JSR      PC,GPWR
1227 005734 004767 177574      :GET SECOND JOINT MOMENT ("POWER")
1228 005744 172437 000000C      AC0,0,SJM
1229 005744 174037 000000C      STF      LDF
1230
1231 005750 112237 000001 000000C      MOVB   *1,0,ICN
1232 005756 004767 000005 000000C      JSR      PC,GHN
1233 005762 172437 000000C      LDF
1234 005766 174037 000004 000000C      STF      *2,0,ICN
1235 005772 112237 000002 000000C      MOVB   *DCV,AC0
1236 006600 004767 000006 000000C      JSR      PC,GHN
1237 006604 172437 000000 000000C      LDF
1238 006610 172537 000000 000000C      STF      *DCV,AC0
1239 006614 171109 000000 000000C      MULF   *COV,AC1
1240 006616 172437 000000 000000C      LDF
1241 006622 173001 000000 000000C      SUBF   *SJM,AC0
1242 006624 174037 000000 000000C      STF
1243 006630 000207
1244
1245 006632 013737 000000 000000C      CPT:          PTR,0,PTC
1246 006632 013737 000000 000000C      MOV      *NCYC,0,MCYF
1247 006640 013737 000000 000000C      MOV      PC,FLUT
1248 006646 004767 176059      JSR      *NODE
1249 006652 005737 000000 000000C      TST
1250 006656 001462
1251 006660 004767 176272      BEQ      *NO
1252 006664 005737 000000 000000C      TST
1253 006670 001462
1254 006672 004767 176322      BEQ      *YES
1255 006676 004767 177100      TST
1256
1257 006102 000207
1258
1259 006104 012765 006116      TJC:          MOV      *TJCA,R5
1260 006110 004767 000000 000000C      JSR      PC,GPOUT
1261 006114 000207
1262 006116 000002 0005076' TJCA:      RTS      PC
                                         2,SJM,NB
                                         PC
                                         :RETURN DATA TO HOST
                                         ;COMPUTE RMS DATA

```

1264  
 1265       ;VOLTMETER SIMULATION ROUTINE.  
 1266       ;IN THIS STATE THE SLAVE PROCESSOR WILL PRETEND TO BE A  
 1267       ;SAMPLING VOLTMETER AND MEASURE/COMPUTE/DISPLAY AS FAST  
 1268       ;AS IT CAN. PRIOR TO THIS MODE BEING ENTERED, VARIOUS  
 1269       ;PARAMETERS NEED TO BE SET UP (AS IN THE NORMAL COMPUTER-  
 1270       ;CONTROLLED SITUATION). THIS MODE PROCEEDS INDEPENDENT  
 1271       ;OF ANY HOST CONTROL, ALTHOUGH THE HOST CAN ABORT THE MODE  
 1272       ;AND THE SLAVE WILL REVERT TO THE INPUT-AWAITING MODE.

1273 006124	013737	0000000C 0000000C		MOV	<del>00PTR,00PTC</del>	:POINTS TO COMPUTE ON
1274 006124	013737	0000000C 0000000C		MOV	<del>00NCYC,00NCYF</del>	:CYCLES TO COMPUTE
1275 006132	013737	0000000C 0000000C		JSR	<del>PC,POARD</del>	:DO MEASUREMENT
1276 006140	004767	174414		CMP	<del>00DSTYLE,00</del>	:NONE
1277				BEQ	408	
1278 006144	023727	0000000C 00000000		CMP	<del>00DSTYLE,01</del>	:CHANNEL 1
1279 006152	001427	0000000C 00000001		CMP	<del>00DSTYLE,02</del>	:COMPUTE VIA DFT
1280 006154	023727	0000000C 00000001		BEQ	508	:CHANNEL 2
1281 006162	001424	0000000C 00000002		CMP	<del>00DSTYLE,03</del>	:COMPUTE VIA DFT
1282 006164	023727	0000000C 00000002		BEQ	508	:CHANNEL 3
1283 006172	001420	0000000C 00000003		CMP	<del>00DSTYLE,04</del>	:COMPUTE VIA DFT
1284 006174	023727	0000000C 00000003		BEQ	508	:VOLTAGES
1285 006202	001414	0000000C 00000004		CMP	<del>00DSTYLE,05</del>	:COMPUTE VIA RMS
1286 006204	023727	0000000C 00000004		BEQ	908	:TWO-CHANNEL AND PHASE
1287 006212	001466	023727		CMP	<del>00DSTYLE,06</del>	
1288 006214	001466	0000000C 00000005		BEQ	908	
1289 006222	001556			HALT		
1290 006224	000090			JMP	RDCODE	
1291 006226	000167	172602		VMDISP		
1292						
1293 006232	000734					
1294				408:	BR	
1295						
1296 006234	013700	0000000C		508:		
1297 006240	010037	0000000C		MOV	<del>00DSTYLE,R0</del>	:DISPLAY MODES 1,2,3 (SINGLE CHANNEL)
1298 006244	005337	0000000C		MOV	<del>R0,00CHNO</del>	:STYLE = CHANNEL NUMBER
1299 006250	006137	0000000C		DEC	<del>00DSTYLE,01,2,3</del>	:LOAD AS (1,2,3)
1300 006254	006137	0000000C		ROL	<del>00CHNO</del>	:BUT CONVERT TO (0,1,2)
1301 006260	062700	000060		ADD	<del>00CHNO</del>	:THEN (0,2,4)
1302 006264	110037	0000000C		MOVB	<del>00,R0</del>	:CONVERT (1-3) TO ASCII TO FOOL
1303 006270	004767	176266		JSR	<del>00,ICN</del>	:INPUT MODE TO EXPECT CHANNEL N
1304 006274	004767	176056		JSR	<del>PC,FLOT</del>	:LOAT DATA
1305 006300	004767	176114		JSR	<del>PC,WINDOW</del>	:MULTIPLY BY WINDOW
1306 006304	004767	176300		JSR	<del>PC,BAVER</del>	:AVERAGE DATA
1307 006310	172337	0000000C		LDF	<del>PC,GDFTD</del>	:COMPUTE DATA
1308 006314	012700	0000000C		MOV	<del>00VDFT,AC0</del>	:ACCESS VOLTAGE
1309 006320	063700	0000000C		ADD	<del>00MSVI,R0</del>	
1310 006324	174019			STF	<del>AC0,(R0)</del>	
1311 006326	172437	0000000C		LDF	<del>00PDFT,AC0</del>	:ACCESS PHASE
1312 006332	012700	0000000C		MOV	<del>00PHASI,R0</del>	
1313 006336	063700	0000000C		ADD	<del>00CHNO,R0</del>	
1314 006342	174019			STF	<del>AC0,(R0)</del>	
1315 006344	172437	0000000C		LDF	<del>00DIST,AC0</del>	
1316 006350	012700	0000000C		MOV	<del>00DIST1,R0</del>	
1317 006354	063700	0000000C		ADD	<del>00CHNO,R0</del>	
1318 006360	174019			STF	<del>AC0,(R0)</del>	
1319 006362	004767	000560		JSR	<del>PC,CHDIS</del>	
1320 006366				BR	<del>1008</del>	

```

1322 006376 112737 000061 0000000C 808:      ;DISPLAY MODE 4 (THREE CHANNELS)
1323 006376 112737 000061 0000000C 808:      ;CHANNEL 1
1324 006376 004767 175520 0000000C 808:      ;FLOAT DATA
1325 006402 004767 175750 0000000C 808:      ;MULTIPLY BY WINDOW
1326 006406 004767 176006 0000000C 808:      ;AVERAGE DATA
1327 006412 004767 176564 0000000C 808:      ;GET VOLTAGE
1328 006416 172437 0000000C 808:      ;ACCESS VOLTAGE
1329 006422 174937 0000000C 808:      ;AC0, @, RMSV1
1330 006426 112737 0000062 0000000C 808:      ;STF
1331 006434 013737 0000000C 0000000C 808:      ;JSR, ICN
1332 006442 013737 0000000C 0000000C 808:      ;PC, FLOT
1333 006450 004767 175446 0000000C 808:      ;JSR, WINDOW
1334 006454 004767 175676 0000000C 808:      ;JSR, BAVER
1335 006460 004767 175734 0000000C 808:      ;JSR, RMSV
1336 006464 004767 176512 0000000C 808:      ;JSR, AC0
1337 006470 172437 0000000C 808:      ;LDF
1338 006474 174937 0000000C 808:      ;AC0, @, RMSV2
1339 006509 112737 0000063 0000000C 808:      ;STF
1340 006506 013737 0000000C 0000000C 808:      ;JSR, ICN
1341 006514 013737 0000000C 0000000C 808:      ;@, PTR, @, PTC
1342 006522 004767 175374 0000000C 808:      ;JSR, NCYC, @, NCYF
1343 006526 004767 175624 0000000C 808:      ;PC, FLOT
1344 006532 004767 175662 0000000C 808:      ;JSR, WINDOW
1345 006536 004767 176449 0000000C 808:      ;JSR, BAVER
1346 006542 172437 0000000C 808:      ;JSR, RMSV
1347 006546 174937 0000000C 808:      ;JSR, AC0
1348 006552 004767 001010 0000000C 808:      ;STF
1349 006556 000465 0000000C 808:      ;JSR, CH4DIS
1350                                BR 1008:      ;BR
1351                                ;DISPLAY MODE 5 (DUAL CHANNEL)
1352 006560 112737 000061 0000000C 908:      ;AC0, @, ICN
1353 006566 004767 175330 0000000C 908:      ;PC, FLOT
1354 006572 004767 175560 0000000C 908:      ;JSR, WINDOW
1355 006576 004767 175616 0000000C 908:      ;JSR, BAVER
1356 006602 004767 176056 0000000C 908:      ;JSR, GDFT
1357 006606 172437 0000000C 908:      ;JSR, VDFT, AC0
1358 006612 174937 0000000C 908:      ;LDF
1359 006616 172437 0000000C 908:      ;AC0, @, RMSV1
1360 006622 174937 0000000C 908:      ;STF
1361 006626 112737 000062 0000000C 908:      ;AC0, @, PHASI
1362 006634 013737 0000000C 0000000C 908:      ;MOV, ICN
1363 006642 013737 0000000C 0000000C 908:      ;@, PTR, @, PTC
1364 006650 004767 175246 0000000C 908:      ;JSR, NCYC, @, NCYF
1365 006654 004767 175476 0000000C 908:      ;PC, FLOT
1366 006660 004767 175534 0000000C 908:      ;JSR, WINDOW
1367 006664 004767 175774 0000000C 908:      ;JSR, BAVER
1368 006670 172437 0000000C 908:      ;JSR, GDFT
1369 006674 174937 0000000C 908:      ;LDF
1370 006700 172437 0000000C 908:      ;AC0, @, RMSV2
1371 006704 174937 0000000C 908:      ;STF
1372 006710 172437 0000000C 908:      ;AC0, @, PHAS2
1373 006714 172537 0000000C 908:      ;LDF
1374 006720 173001 0000000C 908:      ;SUBF
1375 006722 174937 0000000C 908:      ;AC0, @, RPHAS
1376 006726 004767 001032 0000000C 908:      ;PC, CH6DIS
1377                                JSR 1008:      ;DISPLAY DATA
                                         ;AND AGAIN
                                         VMDISP

```

```

1380          ;DISPLAY A/D CONVERSION COUNT AND ERROR COUNT.
1381          ;(1FF DSTYLE, EO, 0):
1382          ;FORMAT: A/D CONV=***** ERR=*****
1383          CH0DIS: TST    0*DSyle      ;RIGHT DISPLAY STYLE?
1384 006736 005737 0000000C             BEQ    103    YES
1385 006742 001401                   RTS    PG     NO
1386 006744 000207                   NOV    *CHERRY, R0   LOAD DISPLAY DELIMITERS
1387 006746 012700 0000000C             NOV    *CH0BLK, RI
1388 006752 012701 007114*
1389          208:    NOVB   (R0)+  ;INTO RAM BUFFER
1390 006756 112126
1391 006760 001376
1392          208:    NOVB   (R1)+, (R0)+ ;INTO RAM BUFFER
1393 006762 013701 0000000C             NOV    R9    ;ENCODE CONVERSION COUNT...
1394 006766 006700                   SXT    R9
1395 006776 012702 00000005             NOV    R5, R2
1396 006774 012703 0000017G             NOV    *CHERRY+15, .R3
1397 007000 071027 0000012             DIV    *10, .R0
1398 007004 062701 0000060              ADD    *60, .R1
1399 007010 110143                   MOVB   R1, -(R3)
1400 007012 010001                   NOV    R0, R1
1401 007014 006700                   SXT    R0
1402 007016 077210                   S0B   R2, 308
1403          408:    NOVB   R1ICE, R1 ;ENCODE ERROR COUNT...
1404 007020 013701 0000000G             NOV    R0
1405 007024 006700                   SXT    R6
1406 007026 012702 00000005             NOV    *5, R2
1407 007032 012703 0000031C             NOV    *CHERRY+25, .R3
1408 007036 071027 0000012              DIV    *10, .R0
1409 007042 062701 0000060              ADD    *60, .R1
1410 007046 110143                   MOVB   R1, -(R3)
1411 007050 010001                   NOV    R0, R1
1412 007052 006700                   SXT    R9
1413 007054 077210                   S0B   R2, 408
1414          508:    NOVB   *CH6ER, R5 ;...AND OUTPUT ENTIRE STRING.
1415 007056 012705 007070*             MOV    JSR    PC, WRITE
1416 007062 004767 001162                   RTS    PG
1417 007066 009207
1418          508:    NOVB   WORD 2, CHERRY, CH0LA, DLUN
1419 007070 000002 0000000C 007100*  CH0ER: 2, CHERRY, CH0LA, DLUN
1420 007076 0090000C
1421 007100 000031                   CH0LA: WORD 25,
1422 007102 00000002 0000000C 007112*  CH0ES: 2, CHERRY, CH0LB, DLUN
1423 007110 0000000C
1424 007112 000017 0115 101 057 CH0BLK: .ASCIZ <15>, "A/D CONV= 0 ETR= 0"
1425          103:    NOVB   WORD 15,
1426 007117 104 103
1427 007122 117 116 126
1428 007125 975 040 040
1429 007130 040 040 060
1430 007133 040 105 122
1431 007136 122 075 040
1432 007141 040 040 040
1433 007144 060 000 EVEN

```



LIVH MACRO M1113 23-JUL-82 14:13 PAGE 31

1478	007366	000004	000000C	000005G	CH1A:	4,RMSV1,CHEERY+5.,TWO,TWO
1479	007374	010164'	010164'			
1479	007409	000004	000000C	000016G	CH1B:	4,PHAS1,CHEERY+14.,THREE,ONE
1480	007406	010166'	010162'			
1480	007412	000004	000000C	000027G	CH1C:	4,DIST1,CHEERY+23.,ONE,TWO
1481	007426	010162'	010164'			
1481	007424	000004	000000C	000005G	CH2A:	4,RMSV2,CHEERY+5.,TWO,TWO
1482	007432	010164'	010164'			
1482	007436	000004	000000C	000016G	CH2B:	4,PHAS2,CHEERY+14.,THREE,ONE
1483	007444	010166'	010162'			
1483	007450	000004	000000G	000027G	CH2C:	4,DIST2,CHEERY+23.,ONE,TWO
1484	007456	010162'	010164'			
1484	007462	000004	000000C	000005G	CH3A:	4,RMSV3,CHEERY+5.,TWO,TWO
1485	007470	010164'	010164'			
1485	007474	000004	000000C	000016G	CH3B:	4,PHAS3,CHEERY+14.,THREE,ONE
1486	007502	010166'	010162'			
1486	007506	000004	000000C	000027G	CH3C:	4,DIST3,CHEERY+23.,ONE,TWO
1487	007514	010162'	010164'			
1487	007520	000002	000000C	007530'	CH1ER:	2,CHEERY,CHILA,DLUN
1488	007526	000000C				
1488	007530	000034				
1489	007532	015	061			
1490	007532	015	061			
	007535	126	075	075	CHIBLK:	.ASCII2 <16>*1=V=+**.** P=+**.** D=**.**
	007540	043	043	053		
	007543	043	043	056		
	007546	120	075	053		
	007551	043	043	043		
	007554	056	043	040		
	007557	104	075	043		
	007562	056	043	043		
	007565	060				
						. EVEN

```

1493
1494
1495
1496 0075566 023727 000000C 0000004 CH#DIS: CMP #DSTYLE, #4 :RIGHT DISPLAY STYLE?
1497 0075566 0000007 BEQ 10$ :YES
1498 0075574 001491 RTS :NO
1499 0075566 0000007 PC :LOAD DISPLAY DELIMITERS
1500 0076000 012700 000000C MOV :CHERRY,R0
1501 0076004 012701 007726 MOV :CH4BLK,R1
1502 0076109 112126 (R1)+,(R0)+ :INTO RAM BUFFER
1503 007612 001376 BNE 20$ :CONVERT CHANNEL ONE VOLTAGE TO ASCII
1504 007614 012765 007656 MOV :CH4A,R5
1505 007620 004767 000000C JSR :PC,CONVRT
1506 007624 012703 007670 MOV :CH4B,R5
1507 007630 004767 000000C JSR :PC,CONVRT
1508 007634 012705 007702 MOV :CH4C,R5
1509 007640 004767 000000C JSR :PC,CONVRT
1510 007644 012705 007714 MOV :CH4ER,R5
1511 007650 004767 000314 :VALUE FILL DISPLAY
1512 007654 000207 RTS :PC,WRITE
1513
1514 007656 000004 000000C 0000004G CH4A: 4,RMSV1,CHERRY+4,,ONE,TWO
1515 007664 010162, 010164, 000004 0000015G CH4B: 4,RMSV2,CHERRY+13,,ONE,TWO
1516 007676 000004 000000C 00000015G CH4C: 4,RMSV3,CHERRY+22,,ONE,THREE
1517 007702 000004 0000004 0000026G CH4E: 2,CHERRY,CH4LA,DLJN
1518 007714 000002 000000C 007724 CH4ER: CH4LA: .WORD 2B.
1519 007724 000004
1520 007726 015 126 061 CH4BLK: .ASCIZ <15>*V1=+*,** V2=+*,** V3=+*,*** 1521
007731 075 053 043
007734 056 043 043
007737 040 126 062
007742 075 053 043
007745 056 043 043
007750 049 126 063
007753 075 053 043
007756 056 043 043
007761 043 069 .EVEN

```

```

1523          ;SUBROUTINE TO DISPLAY DATA ON VOLTMETER
1524          ;FORMAT 5: V1=+*.*** V2=+*.*** P=+**.**
1525          ;RIGHT DISPLAY STYLE?
1526 007764  023727  000000C 0000005  CH5DIS: CMP    *DSTYLE, #5
1527 007764  001461  BEQ    108
1528 007772  012766  000000C 0000005  RTS    PC
1529 007774  012766  000000C 0000005  RT5    PC
1530 007776  012766  000000C 0000005  RT5    PC
1531 010662  012761  010124'      108:  MOV    *CH5BLK,R0
1532 010666  112126  (R0)+     208:  MOVB   *CH5BLK,R1
1533 010610  001376  (R0)+     BNE    208
1534 010612  012705  010054'      MOV    *CH5A,R5
1535 010616  004767  000000C 0000005  JSR    PC,CONVRT
1536 010622  012705  010066'      MOV    *CH5B,R5
1537 010626  004767  000000C 0000005  JSR    PC,CONVRT
1538 010632  012765  010100'      MOV    *CH5C,R5
1539 010636  004767  000000C 0000005  JSR    PC,CONVRT
1540 010642  012705  010112'      MOV    *CH5E,R5
1541 010646  004767  000116'      JSR    PC,WRITE
1542 010652  0000207     RTS   PC
1543          ;CONVERT CHANNEL ONE VOLTAGE TO ASCII
1544 010654  000004  000000C 0000004C CH5A: 4.RHSV1,CHERRY+4.,ONE,THREE
1545 010662  010162'      010166' 000004  0000016C CH5B: 4.RHSV2,CHERRY+14.,ONE,THREE
1546 010666  000004  000000C 0000000C 0000027C CH5C: 4.RPIAS,CHERRY+23.,TWO,ONE
1547 010100  000004  000000C 00000002  010122' CH5ER: 2,CHERRY,CH5LA,DLUN
1548 010106  010164'      010162' 0000020  0000000C 0000034  CH5LA: .WORD 28.
1549          ;CONVERT CHANNEL TWO VOLTAGE TO ASCII
1550 010124  015      126    061  CH5BLK: .ASCIZ <15>*V1=+*.*** V2=+*.*** P=+**.**
1551          ;CONVERT CHANNEL THREE VOLTAGE TO ASCII
1552 010127  075    053    043
1553 010162  0000001     043
1554 010164  0000002     043
1555 010166  0000003     043
1556 010124  015      126    061  CH5BLK: .ASCIZ <15>*V1=+*.*** V2=+*.*** P=+**.**
1557 010127  075    053    043
1558 010132  056    043    043
1559 010135  043    040    126
1560 010140  062    075    053
1561 010143  043    056    043
1562 010146  043    043    040
1563 010151  120    075    053
1564 010154  043    043    056
1565 010157  043    000    EVEN
1551          ;ONE: .WORD 1
1552          ;TWO: .WORD 2
1553          ;THREE: .WORD 3

```

```

;SUBROUTINES TO HANDLE CHARACTER OUTPUT:
;CALL WRITE (IBUFF,ILEN,DLUN)---ASCII STRING
;CALL PRINT (NUMBER,ILEN,DLUN)---DECIMAL NUMBER

1557
1559
1560      010179  010560  000002          WRITE:    MOV    2(R5),R0      ;SET UP START ADDRESS
1561      010174  011750  000004          MOV    04(R5),R1      ;SET UP LENGTH
1562      010220  003404          BLE   30$              ;IF NEG OR ZERO, ERROR
1563      010200  003404          MOVB  (R0)+,R2      ;GET NEXT CHARACTER
1564      010202  112062          JSR    PG,PUTCHR
1565      010204  004767  000004          SOB    R1,10$      ;LOOP ON CHAR COUNT
1566      010210  077104          RTS    PC
1567      010212  000207          RTS    PC

1568      010214  022775  000002  000006  PUTCHR: CNP    *2,06(R5)  ;WHICH DEVICE?
1569      010222  001407  000200  177564          BEQ    CHCHR
1570      010222  001407  000200  177564          BIT    *200,*DLVXSR
1571      010224  032737  000200  177564          BEQ    PUTCHR
1572      010232  001770  110237  177566          BEQ    R2,*DLVXBF
1573      010234  110237  177566          NOVB  PC
1574      010240  000207          RTS    PC

1575      010242  032737  000200  176504  CHCHR:  BIT    *200,*CHXSR
1576      010250  001774  110237  176506          BEQ    CICIR
1577      010250  001774  110237  176506          NOVB  R2,*CHXBF
1578      010252  110237  176506          RTS    PC

1579      010256  000207          PRINT:   *PRINT A DECIMAL INTEGER
1580
1581
1582      010260  012703  000000G          PRINT:   *STRING,R3  ;SETUP OUTPUT BUFFER
1583      010260  012703  000000G          MOV    4(R5),R2  ;GET HOW MANY DIGITS
1584      010264  011502  000004          MOV    02(R5),R1  ;GET NUMBER TO PRINT
1585      010270  011501  000002          TST    R1
1586      010274  005370  002002          BGE    58          ;WORRY IF NEGATIVE
1587      010276  002002          NEG    R1
1588      010300  005401          DEC    R2
1589      010302  005302          SXT    R0
1590      010304  006760          5$:   DIV    *10*,R0      ;ONE LESS TO PRINT
1591
1592      010306  071027  000012          10$:   ADD    *60*,R1      ;AND EXTEND TO 32 BITS
1593      010312  062701  000060          NOV    R1,(R3)+  ;DIVIDE BY 10
1594      010316  010123          MOV    R0,R1      ;CONVERT REMAINDER TO ASCII
1595      010320  010001          SXT    R0
1596      010322  006700          DEC    R2
1597      010324  005302          BGT    10$      ;AND STORE IN STRING
1598      010326  003367          TST    R0
1599
1600      010330  005775  000002          20$:   MOV    *2(R5)  ;CHECK SIGN
1601      010334  002004  000004          BGE   20$      ;CHARACTERS TO PRINT
1602      010336  012702  000055          NOV    *55,R2  ;ADDRESS OF STRING (MSD)
1603      010342  094767  177646          JSR    PG,PUTCHR
1604
1605      010346  010301  000004          30$:   MOV    4(R5),R1  ;PRINT DIGIT
1606      010352  014302          NOV    -(R3),R2
1607      010354  004767  177634          JSR    R1
1608      010360  005301          DEC    R1
1609      010362  003373          BGT    30$      ;DONE YET?
1610      010364  000207          RTS    PC
1611
1612
1613      000001          .END

```

(2) MODULE A32D - Subroutine to control ADAC A/D converters

Calling Sequence:

CALL A32D (BUFFER, GAINS, NWORDS, ADS, GATED, CHNO, ISW)

where the arguments are:

BUFFER	Integer array to receive contiguous sampled data
GAINS	Integer array containing gains of individual A/D converters
NWORDS	Number of samples to digitize per channel
ADS	A/D setup word containing which A/D's in use; which to interrupt from
GATED	Integer variable specifying presence (1) or absence (0) of DRV11/toneburst gate
CHNO	Integer array containing port (channel) numbers for each channel
ISW	Integer variable for returned status

Internal Summary:

<u>(LABEL)</u>	<u>(FUNCTION)</u>	<u>(PAGE)</u>
<u>SETUP ROUTINES</u>		
A32D	Argument transfer; parameter checking	4
SETDMA	Load DMA bus address and word count registers; load pseudo-register for DMA control	4
SETA2D	Load pseudo-register for A/D control; load DMA pseudo-registers into control status registers	6
<u>MEASUREMENT ROUTINE</u>		
READY	Prepare for measurement	7
GO	Check for pulsed-system gate	7
GOCW	Load A/D pseudo registers into actual control status registers; wait for completion (interrupt)	7

ADISR	Dummy interrupt service routine	8
DMISR	Functional interrupt service routine	8

```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57

.TITLE A32D
.IDENT /07/
;SUBROUTINE TO LOAD AN INTEGER DATA ARRAY WITH DATA FROM UP
;TO THREE ADAC-1012 A/D CONVERTERS, RUNNING SIMULTANEOUSLY.
;AUTHOR: RICHARD E. SCOTT, JR. (DECEMBER 1981)
;FORTRAN CALL-
;CALL A32D (BUFFER,CAINS,NWORDS,ADS,CATED,CHNO,ISW)

;"BUFFER" IS AN INTEGER ARRAY TO CONTAIN CONTIGUOUS DATA FROM
;UP TO THREE CHANNELS OF DMA'ED DATA FROM THE A/D CONVERTERS.
;EACH DATA WORD IS A 12-BIT INTEGER. FOR N WORDS PER CHANNEL...
; CHANNEL 1 DATA IN BUFFER(1) TO BUFFER(N)
; CHANNEL 2 DATA IN BUFFER(N+1) TO BUFFER(2N)
; CHANNEL 3 DATA IN BUFFER(2N+1) TO BUFFER(3N)

;"CAINS" ARE PROGRAMMABLE GAIN CODES (1,2,4,B) OR (1,2,5,10)
;FOR A/D CONVERTERS (AS DEFINED BY HARDWARE STRAPPING).
;CAINS IS AN INTEGER ARRAY.

;"NWORDS" IS THE NUMBER OF WORDS PER CHANNEL TO BE DMA'ED
;INTO MEMORY FROM THE A/D CONVERTERS.

;"ADS" IS THE A/D SETUP ARRAY WHERE:
; ADS(1) = A/D'S USED (*1=1) + (*2=2) + (*3=4) (ALL=7)
; ADS(2) = DMA TO GET INTERRUPT FROM (1,2,3)
; AGAIN, WITH ADS(1) = 3 ONE CAN USE TWO A/D CARDS ONLY WITHOUT
; CONCERN FOR THE SUBROUTINE ADDRESSING THE NONEXISTENT THIRD
; A/D CARDS.

;"CATED" FLAGS THE PRESENCE OF PULSE/GATING MECHANISMS.
;CATED = 1 MEANS USE PARALLEL 1,0 CARD TO FLAG START OF PULSE.
;CATED = 0 MEANS DON'T.
;THE LATTER CASE MEANS THAT A PARTICULAR BACKPLANE DOES NOT
;NEED TO HAVE A DRV-11 INSTALLED TO AVOID TRAPS DUE TO ADDRESSING
;UNFEEDED AND NONEXISTENT CARDS. HOWEVER, ONE MAY FIND OCCASIONAL
;PHASE ERRORS BECAUSE OF CLOCK PULSES OCCURRING DURING THE SETUP
;OF THE A/D REGISTERS. ONE TECHNIQUE USED IS TO REQUIRE THAT THE
;HOST PROCESSOR TURN OFF THE SAMPLING CLOCK UNTIL THE REGISTERS
;ARE ALL PREPARED.

;"CHNO" IS AN ARRAY OF THREE CHANNEL NUMBERS, CHANNEL IN THE SENSE
;OF THE SIXTEEN POSSIBLE MULTIPLEXED PORTS TO EACH A/D CONVERTER.

;"ISW" IS STATUS OF TRANSFER (0 = AOK)
;ISW = -1: A/D #1 ERROR (THESE 6 ERRORS ARE ADDITIVE)
;ISW = -2: A/D #2 ERROR (DITTO...)
;ISW = -4: A/D #3 ERROR
;ISW = -10: DMA #1 ERROR
;ISW = -20: DMA #2 ERROR
;ISW = -40: DMA #3 ERROR (...RANGING -1 TO -77)
;ISW = -100: PARAMETER ERROR (FROM THIS PROGRAM: A/D SETUP ERROR)
;ISW = -101: PARAMETER ERROR (FROM CALLING PROGRAM: TOO MANY POINTS)
;***** ALL ARGUMENTS ARE INTEGERS ****

```

## 59 ;HARDWARE REGISTER DEFINITIONS

```

60
61
62
63 164040 ;REGISTERS FOR THE FIRST A/D CONVERTER (A):
64 164042 ADCSRA = 164040 ;A/D CONTROL STATUS REGISTER
65 172410 ADDBRA = 164042 ;A/D DATA BUFFER
66 172412 DNWCRRA = 172410 ;DMA WORD COUNT REGISTER
67 172414 DMCARA = 172412 ;DMA BUS ADDRESS REGISTER
68 172416 DNIMUXA = 172414 ;DMA STATUS REGISTER
69 172416 ;DMA CHANNEL DATA

70
71 164050 ;REGISTERS FOR THE SECOND A/D CONVERTER (B):
72 164052 ADCSRB = 164050 ;A/D CONTROL STATUS REGISTER
73 172426 ADDBRC = 164052 ;A/D DATA BUFFER
74 172422 DNWCRB = 172426 ;DMA WORD COUNT REGISTER
75 172424 DMCARB = 172422 ;DMA BUS ADDRESS REGISTER
76 172426 DNIMUXB = 172424 ;DMA STATUS REGISTER
77 172426 ;DMA CHANNEL DATA

78
79 164060 ;REGISTERS FOR THE THIRD A/D CONVERTER (C):
80 164062 ADCSRC = 164060 ;A/D CONTROL STATUS REGISTER
81 172430 ADDBRC = 164062 ;A/D DATA BUFFER
82 172432 DNWCRC = 172430 ;DMA WORD COUNT REGISTER
83 172434 DMCARC = 172432 ;DMA BUS ADDRESS REGISTER
84 172436 DNIMUXC = 172434 ;DMA STATUS REGISTER
85 172436 ;DMA CHANNEL DATA

86
87
88 164070 ;A/D CONTROL STATUS REGISTER (ADCSR*):
89 164072 ;BIT 15: ERROR BIT (SET FOR CONVERSION TIMING ERROR)
90 164074 ;BIT 14: INTERRUPT ON ERROR (0)
91 164076 ;BIT 12-3: NOT USED (EXTENDED CHANNEL ADDRESS)
92 164078 ;BIT 11-8: ADDRESS OF CHANNEL (#DD)
93 164080 ;BIT 7: A/D DONE FLAG (SET AT END OF CONVERSION)
94 164082 ;BIT 6: INTERRUPT ENABLE (0)
95 164084 ;BIT 5: OPTIONAL ENABLE (0)
96 164086 ;BIT 4-3: PROGRAMMABLE GAIN (AS INPUT)
97 164088 ;BIT 2: SEQUENTIAL ENABLE
98 164090 ;BIT 1: EXTERNAL ENABLE (1)
99 164092 ;BIT 0: A/D CONVERSION GO BIT (0)

100
101
102
103
104
105
106
107
108 ;DATA REGISTER (ADDRH*):
;BITS 12-15: NOT USED (1111)
;BITS 00-11: DATA
;DMA WORD COUNT REGISTER (DNWCR*):
;BITS 12-15: NOT USED (1111)
;BITS 00-11: ONE'S COMPLEMENT OF WORDS
;DMA BUS ADDRESS (DMCAR*):
;BITS 00-11: WORD ADDRESS

```

```

110          ;DMA CONTROL STATUS REGISTER (COMCSR*):
111          ;BIT 15: ERROR BIT (SET FOR ANY ERROR)
112          ;BIT 14: NONEXISTENT MEMORY ERROR
113          ;BIT 13: ATTENTION BIT (EXTERNAL)
114          ;BIT 12-8: NOT USED (0)
115          ;BIT 7: READY FLAG (SET WHEN DMA DONE)
116          ;BIT 6: INTERRUPT ENABLE (0)
117          ;BIT 5-4: EXTENDED ADDRESS BITS
118          ;BIT 3-1: NOT USED (0)
119          ;BIT 0: DMA CO BIT (1)

120
121
122          ;REGISTERS FOR THE DRV-11 USED FOR SYNCHRONIZATION IN PULSED
123          ;SYSTEMS.
124          ;DRCSRH = 167770
125          ;DROUT = 167772
126          ;DRINN = 167774
127
128
129
130          ;VECTOR ADDRESSES:
131          ;130 - A/D COMPLETION (A) OR ERROR
132          ;150 - DMA COMPLETION (A)
133          ;170 - A/D COMPLETION (B) OR ERROR
134          ;210 - DMA COMPLETION (B)
135          ;230 - A/D COMPLETION (C) OR ERROR
136          ;250 - DMA COMPLETION (C)
137
138
139          ;SET PROM = 1 IF VECTOR SPACE IS WITHIN ROM SPACE AND SET
          ;VECTORS REQUIRED. OTHERWISE, VECTORS ARE LOADED DYNAMICALLY.
          ;SET PROM = -1 FOR RAM VARIABLES ASSEMBLED LOCALLY.
          ;PROM=1
140          000001

```

```

142          ;SUBROUTINE INITIALIZATION:
143          ;GRAB THE ARGUMENTS.
144          ;DO SOME CHECKING FOR VALIDITY.
145          A32D:   ;START OF ROUTINE
146          ;IF HAVE, THEN...
147          000000  005775  0000012    TST    #012(R5)
148          000004  001402    BEQ    108
149          000006  005037  167776    CLR    #DRCSR
150          ;CLEAR UP THE DRV-11 STATUS REGISTER
151          ;ADDRESS OF ADS
152          000012  016500  0000010    MOV    10(R5),R0
153          000016  011067  000000C    MOV    (R0),WAD
154          000022  016067  000002    MOV    2(R0),IWD
155          000030  016500  000002    MOV    2(R5),R0
156          000034  017504  000006    MOV    #06(R5),R4
157          000040  005104    COM    R4
158          000042  016502  000014    MOV    14(R5),R2
159          ;ADDRESS OF A/D-S-USED WORD
160          000046  022767  000007    CMP    #7,WAD
161          000054  002414    BLT    PERR
162          000056  022767  000001    CMP    #1,WAD
163          000064  003019    BCT    PERR
164          000066  022767  000003    CMP    #3,IWD
165          000074  002404    BLT    PERR
166          000076  022767  000001    CMP    #1,IWD
167          000104  003404    BLE    SETDMA
168          ;ADDRESS OF DMA INTERRUPT WORD
169          000106  012775  177700    PERR:  MOV    #100, #116(R5)
170          000114  0000207   RTS    PC      ;PARAMETER ERROR
171          ;SET UP THE NECESSARY REGISTERS FOR THOSE A/D-DMAS
172          ;THAT ARE TO BE USED.
173
174          ;FIRST DMA REGISTERS SETUP:
175
176          ;IS THIS DMA ACTIVE?
177          000116  032767  000001    SETDMA: BIT    #1,WAD
178          000124  001424    BEQ    208
179          000126  010437  172410    NOV    R4, #>DMICRA
180          000132  010037  172412    NOV    R0, #>DMICARA
181          000136  012737  0000001   NOV    #1, #>T1
182          000144  012261    NOV    (R2)+,RI
183          000146  010137  172416    NOV    R1, #>DMUXKA
184          000152  0000301   SWAB   RI
185          000154  010137  0660000   NOV    R1, #>ADD
186          ;SHIFT INTO BITS 6-11
187          ;THEN STORE FOR LATER USE
188          ;LOAD INTERRUPT VECTOR (A/D)
189          ;AND PROCESSOR STATUS WORD
190          ;LOAD INTERRUPT VECTOR (DMA)
191          ;AND PROCESSOR STATUS WORD
192          000160  022767  000001    ENDC
193          000166  001063    NOV    GRP    #1,IWD
194          000170  012737  000101    NOV    #340, #132
195          ;NOPE!
196          ;LOAD CSR, IE AND GO
197          ;LOAD ACTIVE ISR VECTOR

```

```

199
200    000176  067500  000006          ADD    06(R5),R0      ! INCREASE BUFFER ADDRESS
201    000202  067500  000006          ADD    06(R5),R0      !(TWICE FOR BYTES)
202    000206  032767  000002          BIT    *2,WAD        !IS THIS DMA ACTIVE?
203    000214  001424          000000C          BEQ    30$          !NOPE!
204
205    000216  010437  172426          MOV    R4,*DMWCRCB
206    000222  010437  172422          MOV    R0,*DMCAHB
207    000226  012737  000001          MOV    R1,*T2
208    000234  012201          000000C          MOV    (R2)+,R1
209    000236  010137  172426          MOV    R1,*DMUXB
210    000242  000301          000000C          SWAB   R1
211    000244  010137  000000C          MOV    R1,*BDD
212
213    000259  022767  000002          EQ    PRIM, *ADISR, @*170
214    000256  001003          000000C          MOV    *340,*172
215
216    000260  012737  000101          EQ    PRIM, *DMISR, @*210
217
218    000259  022767  000002          ENDCCMP,1WD
219    000256  001003          000000C          BNE   30$          !INTERRUPT OFF THIS ONE?
220    000260  012737  000101          MOV    *101,*T2
221
222    000326  010137  172436          IF    EQ    PRIM, *DMISR, @*210
223
224    000266  067500  000006          ENDCCMP,1WD
225    000272  067500  000006          ADD    06(R5),R0      ! INCREASE BUFFER ADDRESS
226    000276  032767  000004          ADD    06(R5),R0      !(TWICE FOR BYTES)
227    000304  001426          000000C          BIT    *4,WAD
228
229    000306  00437   172430          BEQ    SETA2D
230
231    000312  010037  172432          MOV    R4,*DMWCRC
232    000316  012737  000001          MOV    R0,*DMCAHC
233    000324  012201          000000C          MOV    *1,*T3
234
235    000326  010137  172436          MOV    (R2)+,R1
236    000332  000301          000000C          MOV    R1,*DMUXC
237    000334  042701  170377          SWAB   R1
238    000340  010137  000000C          BIC    *170377,R1
239
240    000344  022767  000003          MOV    R1,*CDD
241
242    000352  001003          000000C          IF    EQ    PRIM, *DMISR, @*250
243
244    000354  012737  000101          MOV    *340,*232
245    000344  022767  000003          ENDCCMP,1WD
246    000352  001003          SETA2D
247    000354  012737  000101          BNE   *101,*T3
248
249    000340  010137  000000C          IF    EQ    PRIM, *DMISR, @*250
250

```

252  
 253  
 254  
 255  
 256  
 257  
 258  
 259 000362 016503 000004 SETA2D: MOV 4(R5), R3  
 260 000366 017567 000010 000000C MOV 010(R5), WAD  
 261  
 262  
 263  
 264 000374 012700 000000C 000001 MOV #ADUM,R0  
 265 000400 032767 000001 000000C BIT #1,WAD  
 266 000406 001425 BEQ 30\$  
 267 000410 012300 NOV (R3)+,R0  
 268 000412 006200 ASR R0  
 269 000414 022700 CMP #5,R0  
 270 000420 003602 BGT 20\$  
 271 000422 005300 DEC R0  
 272 000424 005300 DEC R0  
 273 000426 072027 000003 ASH #3,R0  
 274 000432 005100 CON R0  
 275 000434 042700 BIC #177747,R0  
 276 000440 053700 BIS #\*ADD,R0  
 277 000444 010037 NOV R0, #\*ADCNSRA  
 278 000450 012700 MOV #ADCNSRA,R0  
 279 000454 016737 NOV T1, #\*DMCNSRA  
 280  
 281  
 282  
 283 000462 012701 000000C 0000002 MOV #ADUM,RI  
 284 000466 032767 000002 000000G BIT #2,WAD  
 285 000474 001425 BEQ 50\$  
 286 000476 012301 NOV (R3)+,RI  
 287 000500 006201 ASR RI  
 288 000502 022701 CMP #5,RI  
 289 000506 003602 BGT 40\$  
 290 000510 005301 DEC RI  
 291 000512 005301 DEC RI  
 292 000514 072127 000003 ASH #3,RI  
 293 000520 005101 COM RI  
 294 000522 042701 BIC #177747,RI  
 295 000526 053701 BIS #\*BDD,RI  
 296 000532 010137 NOV RI, #\*ADCNSRB  
 297 000536 012701 MOV #ADCNSRB,RI  
 298 000542 016737 NOV T2, #\*DMCNSRB

;SET UP A/D CONVERTER NUMBER ONE IF ACTIVE  
 ;TARGET ADDRESS IF NOT ACTIVE  
 ;IS THIS A/D ACTIVE?  
 ;NOPE  
 ;GET GAIN CODE OF CHANNEL A (1,2,5,10)  
 ;CHANCE TO 0,1,2,5  
 ;CHECK IF 5  
 ;NOPE  
 ;SET 5 TO 4  
 ;THEN TO 3  
 ;SHIFTED INTO BITS 3-4  
 ;SET TO 30,10,00  
 ;SAFETY CLEANUP  
 ;ADD ADDRESS SETTING TO R0  
 ;SET GAIN (ETC) BITS FIRST  
 ;TARGET ADDRESS IF ACTIVE  
 ;ENABLE DMA'S

;SET UP A/D CONVERTER NUMBER TWO IF ACTIVE  
 ;TARGET ADDRESS IF NOT ACTIVE  
 ;IS THIS A/D ACTIVE?  
 ;NOPE  
 ;GET GAIN CODE OF CHANNEL B (1,2,5,10)  
 ;CHANCE TO 0,1,2,5  
 ;CHECK IF 6  
 ;NOPE  
 ;SET 6 TO 3  
 ;SHIFTED INTO BITS 3-4  
 ;SET TO 30,10,00  
 ;ADD ADDRESS SETTING TO RI  
 ;SET GAIN (ETC) BITS FIRST  
 ;TARGET ADDRESS IF ACTIVE  
 ;ENABLE DMA'S

```

300                                     :SET UP A/D CONVERTER NUMBER THREE IF ACTIVE
301    0000550   012702   000000C   0000004   0000000C   508:   NOV      *ADUM,R2          ;TARGET ADDRESS IF NOT ACTIVE
302    0000554   032767   000000C   0000004   0000000C   NOV      *4,WAD           ;IS THIS A/D ACTIVE?
303    0000562   0011425    BIT      *4,WAD           ;NOPE
304    0000564   012302    BEQ      READY             ;GET GAIN CODE OF CHANNEL C (1,2,5,10)
305                                     ;CHANGE TO 0,1,2,5
306    0000566   006292    NOV      (R3)+,R2         ;CHECK IF 6
307    0000570   022792    ASR      R2
308    0000574   003902    CHP      *5,R2           ;NOPE
309    0000576   005302    BGT      608              ;SET 5 TO 3
310    0000609   005302    DEC      R2
311    0000602   072227   0000003   608:   ASH      *3,R2           ;SHIFTED 1 INTO BITS 3-4
312    0000606   005102    CON      R2
313    0000616   042516    BIC      *177747,R2        ;ADD ADDRESS SETTING TO R2
314    0000614   053792   000000C   000000C   BIS      @*CDD,R2          ;SET GAIN (ETC) BITS FIRST
315    0000620   010237   164060    NOV      R2,@ADCSCRC     ;TARGET ADDRESS IF ACTIVE
316    0000624   012702   164060    NOV      @ADCSCRC,R2
317    0000630   016737   000000C   172434    MOV      T3,@ADCSCRC     ;ENABLE DMA S
318
319
320
321                                     ;IS EVERYBODY READY???
322                                     ;INCLUDING TRANSMIT GATE TO DLV-11???
323                                     ;CLEAR STATUS WORD
324    0000636   005067   000000C   000000C   READY:  CLR      ADST             ;CLEAR ERROR STATUS
325    0000642   005075   000016    CLR      @16(R5)
326
327    0000646   0000249    NOP      TST      @12(R5)          ;USE DRV-11 FOR PULSE GATE?
328    0000650   005575   0000012    BEQ      COCW             ;NO
329    0000654   001404    BIT      *1,@DRINN        ;CHECK FOR BIT 0 SET
330    0000656   032737   000001   167774   CO:      BEQ      GO
331    0000654   001774    ...AND WAIT...
332
333                                     ;THE FOLLOWING IS DONE THIS WAY SO THE LOADING IS AS FAST
334                                     ;AS POSSIBLE, THUS ELIMINATING ANY POSSIBLE PHASE ERRORS
335                                     ;BETWEEN THE THREE CHANNELS.
336                                     ;TIME TO LOAD EACH CSR IS ABOUT 4.54 MICROSECONDS.
337                                     ;NOTE THAT IF ACTIVE, THE DESTINATION REGISTERS CONTAIN THE
338                                     ;ADDRESSES OF THE A/D CSR'S; IF NOT ACTIVE, THEY CONTAIN A
339                                     ;DUMMY ADDRESS TO AVOID THE POSSIBILITY OF WRITING TO A
340                                     ;NON-EXISTENT ADDRESS (AND TRAPPING) IF INACTIVE A/D CONVERTERS
341                                     ;ARE NOT IN THE BACKPLANE.
342
343    0000666   012703   0000003    COCW:  NOV      *3,R3          ;START A/D CONVERSIONS
344    0000672   050310    BIS      R3,(R0)
345    0000674   050311    BIS      R3,(R1)
346    0000676   050312    BIS      R3,(R2)
347
348    0000700   0000001    208:   WAIT    TST      ADST             ;WAIT FOR SPECIFIED INTERRUPT
349    0000702   005767   000000C   NOP
350    0000706   000249    BEQ      208              ;...AND WAIT...
351    0000710   001773    RTS      PC
352
353    0000712   0002087    ,RETURN WITH DATA

```

```

355          ;INTERRUPT SERVICE ROUTINE
356          ; DMSR -- DMA SERVICE ROUTINE
357          ; A/DISR -- A/D SERVICE ROUTINE
358          ;WITH SIMULTANEOUS ACTION, ALL SHOULD FINISH AT ONE TIME, THUS
359          ;NO NEED TO GO THROUGH ISR FOR ALL THREE DEVICES.
360          ;(INTERRUPTS DISABLED ON REMAINING DEVICES ANYWAY.)
361
362 000714 000002          ADISR:: RTI      ;DO NOTHING
363
364 000716 012767 000001 000000C DMSR:: MOV      ;1,ADST ::GOOD INTERRUPT: EOR FOR CHANNEL SPECIFIED
365          BIT      ;1,WAD   ;THIS A/D ACTIVE?
366 000724 032767 000001 000000C BEQ      ;1,NOPE
367 000732 001414 001414 001414 TST      ;1,ERRORS FOR CHANNEL A?
368 000734 005737 164040 005737 BEQ      ;1,NO
369 000740 1000003 1000003 BPL      ;1,NO
370 000742 062775 177777 0000016 ADD      ;*-1,016(R5)  ;YES: ERROR -1+
371 000750 005737 172414 1000003 TST      ;1,ERRORS FOR CHANNEL A?
372 000754 1000003 1000003 BPL      ;1,NO
373 000756 062775 177777 0000016 ADD      ;*-1,016(R5)  ;YES: ERROR -10+
374 000764 032767 000002 000000C 299:    BIT      ;1,NOPE!
375 000772 001414 001414 BEQ      ;1,NOPE!
376 000774 005737 164050 005737 TST      ;1,ERRORS FOR CHANNEL B?
377 001000 1000003 1000003 BPL      ;1,NO
378 001002 062775 177777 0000016 ADD      ;*-2,016(R5)  ;YES: ERROR -2+
379 001010 005737 172424 309:    TST      ;1,ERRORS FOR CHANNEL B?
380 001014 1000003 1000003 BPL      ;1,NO
381 001016 062775 177760 0000016 ADD      ;*-2,016(R5)  ;YES: ERROR -20+
382 001024 032767 0000004 000000C 409:    BIT      ;1,NOPE!
383 001032 001414 001414 BEQ      ;1,ERRORS FOR CHANNEL C?
384 001034 005737 164060 005737 TST      ;1,NO
385 001040 1000003 1000003 BPL      ;1,NO
386 001042 062775 177774 0000016 ADD      ;*-4,016(R5)  ;YES: ERROR -4+
387 001050 005737 172434 1000003 TST      ;1,ERRORS FOR CHANNEL C?
388 001054 1000003 1000003 BPL      ;1,NO
389 001056 062775 177740 0000016 ADD      ;*-4,016(R5)  ;YES: ERROR -40+
390 001064 0000002 0000002 609:    RTI      ;IF LT PROM
391
392          ;RAM VARIABLES USED BY A32D
393          WAD: .WORD 0
394          IWD: .WORD 0
395          T1: .WORD 0
396          T2: .WORD 0
397          T3: .WORD 0
398          ADUM: .WORD 0
399          ADST: .WORD 0
400          ADD: .WORD 0
401          BDD: .WORD 0
402          CDD: .WORD 0
403          ENDC: .ENDC
404
405          0000001 .END

```

(3) MODULE CONVRT - Subroutine to convert a floating-point number to an ASCII string.

Calling Sequence:

CALL CONVRT (VALUE, STRING, K, L)

where the arguments are:

VALUE Floating-point variable to be encoded

STRING Byte array to load encoded variable

K Encode with K places to left of decimal point

L Encode with L places to right of decimal point

NOTE: Decimal point (.) and sign (+ or -) are not included in the size of K and L.

```

1   TITLE CONVRT
2   IDENT /01/
3
4   ;SUBROUTINE TO CONVERT A REAL NUMBER (FLOATING) TO
5   ;AN ASCII STRING OF FORMAT "KK.LL" FOR PRINTING
6   ;PURPOSES.
7
8   ;CALL. CONVRT (VALUE, STRING,K,L)
9   ;WHERE VALUE BECOMES AS F(K),L. (FORTRAN)
10  ;AUTHOR: RICHARD E. SCOTT, JR. (NRL-USRD -- AUGUST 1961)

11
12          AC9=Z9
13          AC1=Z1

14          000000
15          000001
16          000000
17          000000 010046
18          000002 010146
19          000004 010246
20          000006 010346
21          0J0010 010446
22
23          0000012 170001
24          000014 016503 000004
25          000020 172475 000002
26          000024 170000
27          000026 003003
28          000030 112723 000005
29          000034 000402
30          000036 112723 000053
31          000042 170600
32          000044 175404
33          000046 177101
34          000050 017504 000006
35          000054 004767 000054
36          000060 112723 000056
37          000064 173001
38          000066 017504 000010
39          000072 171037 000166
40          000076 005304
41          000100 003374
42          000102 172637 000172
43          000106 175401
44          000110 017504 000019
45          000114 004767 000014
46
47          000120 012604
48          000122 012603
49          000124 012602
50          000126 012601
51          000130 012600
52
53          000132 0000267

```

80

```

        CONVRT :          MOV    R0,-(SP)      ;SAVE REGISTERS ON STACK
                           MOV    R1,-(SP)
                           MOV    R2,-(SP)
                           MOV    R3,-(SP)
                           MOV    R4,-(SP)
                           SETF   4(R5),R3      ;SET TO FLOATING MODE
                           MOV    02(R5),AC9      ;ADDRESS OF OUTPUT STRING
                           LDF    CFCC            ;GET VALUE TO PRINT
                           BGT    10$              ;POSITIVE OR NEGATIVE?
                           MOVB   '$',-(R3)+      ;INSERT MINUS SIGN
                           BR    20$              ;LOAD THEM IN STRING
                           MOVB   ','+(R3)+      ;INSERT PLUS SIGN
                           ABSF   AC9             ;ABSOLUTE VALUE
                           STCFI  AC9,R1           ;SAVE INTEGER PART (VOLTS?)
                           LDGIF  R1,AC1
                           MOV    06(R5),R4      ;DIGITS LEFT OF DECIMAL
                           JSR    PC,DECO           ;LOAD THEM IN STRING
                           MOVB   ','+(R3)+      ;AND DECIMAL POINT
                           SUBF   AC1,AC9
                           MOV    010(R5),R4      ;FIND FRACTION PART (MILLIVOLTS)
                           MOVL   0*TEN,AC0         ;CONVERT TO 0**L INTEGER
                           DEC    R4
                           BGT    30$              ;ROUND-OFF
                           ADDF   0*RO,AC9
                           STCFI  AC0,R1
                           MOV    010(R5),R4
                           JSR    PC,DECO
                           MOV    (SP)+,R4
                           MOV    (SP)+,R3
                           MOV    (SP)+,R2
                           MOV    (SP)+,R1
                           MOV    (SP)+,R0
                           RTS    PC

```

```

55 000134    010402      DECO:    MOV      R4,R2
56 000134    010402      ;CHARACTER COUNTER
57 000136    060263      ADD      R2,R3
58 000140    0006700     SXT      R0
59 000142    071027    108:    DIV      "10",R0
60 000146    062701    ;EXTEND VALUE TO 32 BITS
61 000152    110143    ;DIVIDE BY 10
62 000154    010001    ;CONVERT REMAINDER TO ASCII
63 000156    0005302   ;AND STORE IN STRING
64 000160    0033367   ;CONVERT QUOTIENT TO DIVIDEND
65 000162    060403   ;DONE?
66 000164    000207   ;NO
67          0           ;RESET STRING POINTER
68 000166    041040    0000000  TEM:    .FLT2  10.
69 000172    037777    171345  RO:     .FLT2  0.4999
70          0           .END
71          0

```

(4) MODULE DAPP - Subroutine to compute voltage and phase from the real and imaginary parts of the voltage, and to compute the harmonic distortion from a specific number of harmonics.

Calling Sequence:

```
CALL DAPP      (X, N, K, AMP, PHASE, DSTORT, LHARM, IERR)
```

where the arguments are:

X	Floating-point data sequence
N	Length of data sequence
K	Spectral line (number of cycles) to compute
AMP	Resulting total rms voltage
PHASE	Resulting phase in degrees
DSTORT	Harmonic distortion
LHARM	Number of harmonics used to compute distortion
IERR	Execution status

- NOTES:
1. If LHARM = 1, distortion is not computed
  2. If DSTORT = -1, distortion was not computed
  3. If DSTORT = -2, error occurred (AMP = 0)
  4. If IERR = -1, K = N (illegal condition)
  5. If IERR = LHARM, harmonic folds onto dc line

MACRO M1113 20-SEP-82 09:55 PAGE 1  
0049P0

```

1      .TITLE DAPD
2      .IDENT /01/
3      ;AUPPIA.LTN 17-JUL-81 JDG GEORGE (ORIGINATOR)
4      ;DAPD .MAC 62-OCT-81 RE SCOTT (TRANSLATOR)
5
6      ;SUBROUTINE DAPD (X,N,K,AMP,PHASE,DSTOIT,LHARM,IERR)
7
8      ;COMPUTE AMPLITUDE, PHASE, & PERCENT HARMONIC DISTORTION
9      ;FOR THE K-TH SPECTRAL LINE OF THE DFT OF X, A REAL SEQUENCE
10     ;OF LENGTH N.
11     ;UNERSAMPLING OR FOLDING WHERE K CAN EXCEED N/2 IS ALLOWED.
12     ;SUBROUTINES REQUIRED: DGZL (X,N,K,XR,XI)
13
14
15     ;REAL X(N)          !DATA SEQUENCE
16     ;INTEGER N          !SEQUENCE LENGTH
17     ;INTEGER K          !DFT SPECTRAL LINE NUMBER OF INTEREST
18
19     ;ALSO > CYCLES OF TEST SINUSOID IN N SAMPLES
20
21     ;REAL AMP           !K'S MAGNITUDE OF TEST SINUSOID IN N SAMPLES
22     ;REAL PHASE          !K'S PHASE IN DEGREES OF TEST SINUSOID
23
24     ;REAL DSTORT         !K'S HARMONIC DISTORTION OF K-TH SPECTRAL COMPONENT
25     ;REAL DPHASE          !K'S PHASE IN DEGREES OF K-TH SPECTRAL COMPONENT
26
27     ;INTEGER LHARM         !MAX HARMONIC TO INCLUDE IN DISTORTION
28
29     ;INTEGER IEUR          !IF LHARM=0 SKIP DISTORTION CALCULATION; SAVE TIME
30
31     ;IF LHARM<0 NORMAL
32
33     ;DSTOIT = -1 IF (DOD(K,N)=0
34     ;DSTOIT = -2 IF HARMONIC FOLDS ON DC LINE
35
36     ;AC0=%0
37     ;AC1=%1
38     ;AC2=%2
39
40     ;HARLIN = 11.      ;MAXIMUM HARMONICS TO USE
41
42     ;EXPERIENCE SEEMS TO SHOW THAT A MINIMUM OF 9 POINTS ARE
43     ;REQUIRED TO COMPUTE ACCURATELY THE AMPLITUDE AND PHASE OF
44     ;A SINUSOID; FOR DISTORTION, 2*LHARM+1 POINTS ARE REQUIRED.
45
46     ;GET PROH = 1 FOR RAM VARIABLES STORED EXTERNALLY
47     ;SET PROH = 0 FOR RAM VARIABLES STORED INTERNALLY (NORMAL)
48     ;PROH=1
49
50     DAPD:::      NOV
51     00000000 010046
52     00000002 010146
53     00000004 010246
54     00000006 010346
55     00000010 010446
56     00000012 010546

```



```

107      CVAF:          SETF    @10(R5),AC0
108      000144 170001    LDF    STF    ;X0
109      000146 172475    000010    AC0,AC2
110      000152 174062    AC0,AC0
111      000154 1710C0    MULF   LDF    ;X0*X0
112      000156 172575    00002     AC1,AC1
113      000162 171101    MULF   LDF    ;X1
114      000164 172001    ADDF   AC1,AC0
115      000166 172000    ADDF   AC0,AC0
116      000167 054742    PC,PSQRT  ;2 (X0*X0 + X1*X1)
117      000174 012665    JSR    NOV    ;=VDFT
118      000176 174973    (SP)+,R5  ;RESTORE R5
119      00010     STF    AC0,910(R5)
120      000202 170592    TSTF   AC2
121      000204 1700C0    CFCC   REQ    ;WATCH FOR ATAN(X1/0)
122      000206 061420    15$    ;ERROR!
123
124      000210 010546    MOV    R5,-(SP)
125      000212 012705    MOV    *ARCBK,R5
126      000216 004767    000000C  PG,SATAN2
127      000222 172400    R0,AC0
128      000224 171037    000670    MULF   ;CONVERT TO DEGREES
129      000230 005767    000000C  @#ANGC,AC0
130      000234 003001    PSIGN  TST
131      000236 170700    BCT    NOV    ;WAGON WHEELS?
132      000240 012665    NEGF   STF    ;FORWARD
133      000242 174075    0C0012    AC0,12(R5) ;NO--BACKWARD ROTATING
134      000246 000404    BR    CD1S   ;RESTORE R5
135      000246 000404    BR    PDFT
136
137      000250 172637    000700    LDF    @#P90,AC2
138      000254 174275    0009012   STF    AC2,@12(R5)

```

;SET PDFT = 90



APD MACRO M1113 20-SEP-82 09:55 PAGE 3

DATAID HACRO M1113 20-SEP-82 09:55 PAGE 6

(5) MODULE DGZL - Subroutine to perform a discrete Fourier transform on a data sequence using Goertzel's algorithm.

Calling Sequence:

CALL DGZL (X, N, K, XR, XI)

where the arguments are:

X	Floating-point data sequence
N	Length of sequence
K	Spectral line (number of cycles) to compute
XR	Real part of voltage
XI	Imaginary part of voltage

```

1      TITLE DGZL
2      IDENT /01/
3      SUBROUTINE DGZL (X,N,K,XR,XI)
4
5      !REAL X(N)          !DATA SEQUENCE
6      !INTEGER N          !SEQUENCE LENGTH
7      !INTEGER K          !DFT SPECTRAL LINE NUMBER OF INTEREST
8      !ALSO * CYCLES OF TEST SINUSOID IN N SAMPLES
9
10     !REAL XR            !REAL PART OF VOLTAGE
11     !REAL XI            !IMAGINARY PART OF VOLTAGE
12
13     !DGRTZL.FTN -- JD GEORGE (17-JUL-81) (ORIGINATOR)
14     !DGZL .MAC -- RE SCOTT (02-OCT-81) (TRANSLATOR)
15
16     ;COERTZEL'S ALGORITHM
17     ;TO COMPUTE K-TH LINE OF N POINT REAL SEQUENCE X(L)
18     ;REFERENCE: G. COERTZEL
19     ;    "AN ALGORITHM FOR THE EVALUATION OF FINITE TRIG SERIES"
20     ;    AMERICAN MATHEMATICAL MONTHLY, V65, 1958, PP.34-35
21     ;ALSO CHAPTER 24, PP.258-62
22     ;MATHEMATICAL METHODS FOR DIGITAL COMPUTERS VOL. 1
23     ;EDITED BY A. RALSTON & H.S. WILF
24     ; JOHN WILEY & SON, 1967
25
26     ;A DOUBLE PRECISION COERTZEL ROUTINE WHICH IS MORE ACCURATE A
27     ;FASTER THAN "SPDFT.FTN".
28     ; 40% RUNNING TIME OF SPDFT.FTN
29     ; UP TO 612 POINTS AMPLITUDE ERROR < 0.0012%
30     ; PHASE ERROR < 0.00091DEG
31
32     00000000
33     0000001
34     000002
35     000003
36
37
38
39     000001
40
41     000000 010546
42     000002 170011
43     000004 170002
44
45     000006 177075 000006
46     000012 177175 000004
47
48     000016 174401
49     000020 171637 000024
50     000024 172699
51     000026 174637 000000C
52
53     000032 004767
54     000036 174637 000000C
55     000042 172699
56     000044 174637 000000C

```

## DCZL MACRO M1113 23-JUL-82 14:14 PAGE 2

```

38 000050 172437 000000C          LDD      @A0,AC0           ;S0=DSIN(A0)
59 000054 004767 000000C          JSR      PC,SSDSIN
60 000060 174037 000000C          STD      AC0,S0
61                                         CLRD    ULP1             ;ULP1=0.0D0
62 000064 170402 0000002          CLRD    ULP2             ;ULP2=0.0D0
63 000066 170403
64                                         MOV     (SP)+,R5          ;RESTORE ARGUMENT LIST
65 000070 012605
66 000072 017501 0000004          MOV     R5,R1           ;COUNTER N IN RI
67 000076 010100
68 000109 005301
69 000102 006300
70 000104 006300
71 000106 006500 0000002          ADD     2(R5),R0           ;REAL*4 DATA BUFFER
72                                         ADD     R0
73                                         DO 10 L=N,2,-1        ;DO 10 L=X(L)+C1*ULP1-ULP2
74 000112 172437 000000C          LDD      R0
75 000116 171002
76 000120 173003
77 000122 177546
78 000124 172001
79 000126 172702
80 000130 172600
81 000132 077111
82                                         LD      R1,AC0           ;X1=(X1+C0*ULP1-ULP2)/(N)
83 000134 172437 000000C          LD      R1,AC0           ;XR=(X1+C0*ULP1-ULP2)/(N)
84 000140 171602
85 000142 173003
86 000144 177575 0000002          LD      R1,AC0           ;XR=(X1+C0*ULP1-ULP2)/(N)
87 000150 172001
88 000152 177175 0000004          LD      R1,AC0           ;XR=(X1+C0*ULP1-ULP2)/(N)
89 000156 174401
90 000160 176075 0000100
91                                         LD      R1,AC0           ;XR=(X1+C0*ULP1-ULP2)/(N)
92 000164 172437 000000C          LD      R1,AC0           ;XR=(X1+C0*ULP1-ULP2)/(N)
93 000170 171002
94 000172 174401
95 000174 170700
96 000176 176075 0000102          RTS     PC
97                                         RTS     PC
98 000202 000207
99 000204 040511 007732 121041 P1: .FLT4 3.1415926535897932
100 000212 064301
101                                         C0:   .IF EQ PROM
102                                         C1:   .FLT4 0.0
103                                         S0:   .FLT4 0.0
104                                         A0:   .FLT4 0.0
105                                         AC0, @12(R5) .END
106 00000000

```

(6) MODULE GPIOSR - This subroutine controls input and output along the IEEE-488 bus. There are three primary functions:

GPINI Initialize the interface and requisite variables; also set up address to use when group execute trigger (GET) is received.

Calling Sequence:

CALL GPINI (RESET)

where the argument is:

RESET Address of routine to transfer to on GET

GPIN Listen function - read a byte string from the IEEE-488 bus, terminated by a character received with EOI true, or by input buffer overflow.

Calling Sequence:

CALL GPIN (INBUF, MAXCH, NCH)

where the arguments are:

INBUF Byte buffer for input string

MAXCH Maximum number of characters allowed

NCH Number of characters received

GPOUT Talk function - set SRQ and transmit a byte string to the IEEE-488 bus.

Calling Sequence:

CALL GPOUT (OUTBUF, NCH)

where the arguments are:

OUTBUF Byte buffer for output string

NCH Number of characters to output

Also included is an interrupt service routine. The only interrupt always enabled is the group execute trigger (GET) interrupt, which feature is used to effect a program abort or restart function. The bus-in and bus-out interrupts are enabled only when their respective subroutines are called and disabled on completion of transfer.

The stand-alone driver supplied by National Instruments Corp. was not used for two reasons: size (keeping the total program size under 4K) and to implement the special abort function. However, programming this complex interface was not a straightforward exercise.

AD-A126 980

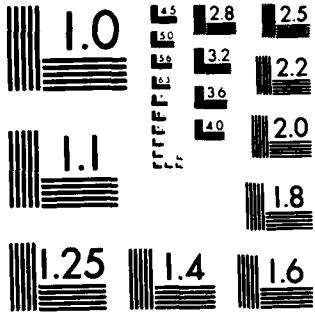
A MICROCOMPUTER-BASED SAMPLING DIGITAL VOLTmeter(U)  
NAVAL RESEARCH LAB WASHINGTON DC R E SCOTT ET AL.  
31 MAR 83 NRL-MR-4872

2/2

UNCLASSIFIED

F/G 14/2

NI



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

```

1   TITLE GPIOSR
2   IDENT /68/
3
4   ;SUBROUTINES FOR HANDLING THE GPIBV-11 INTERFACE
5   ; GPINI -- INITIALIZE BUS
6   ; GPIN  -- INPUT DATA FROM BUS (LISTEN)
7   ; GPOUT -- OUTPUT DATA TO BUS (TALK)
8   ; ISRGP -- INTERRUPT SERVICE ROUTINE
9
10  ;AUTHOR: R. E. SCOTT, JR. (DECEMBER 1981)--V05
11  ;MODIFIED TO USE EOI ON READ (FEBRUARY 1982)--V06
12  ;MODIFIED TO USE R5 ARGUMENT 1/ST (FEBRUARY 1982)--V07
13  ;MODIFIED TO USE TRIGGER (NOT "AB") FOR ABORT--V08
14  ;NRL/UNDERWATER SOUND REFERENCE DETACHMENT
15
16  ;INITIALIZE--CALL GPINI (RESET)
17  ;INITIALIZES THE GPIBV-11 INTERFACE.
18  ;LOADS THE INTERRUPT VECTOR AND PSW (IF ALLOWED).
19  ;LOADS THE "RESET" ADDRESS (GO TO IF "GET" RECEIVED).
20  ;AND ENABLES BOTH INPUT AND OUTPUT INTERRUPTS.
21
22  ;INPUT--CALL GPIN (INBUF,MAXCH,NCH)
23  ;ALLOWS A STRING OF VARIABLE LENGTH "NCH" (BUT LESS THAN
24  ;"MAXCH") TO BE LOADED IN "INBUF". EOI MUST COME WITH LAST BYTE.
25
26  ;OUTPUT--CALL GPOUT (OUTBUF,NCH)
27  ;OUTPUTS A STRING "OUTBUF" OF VARIABLE LENGTH "NCH"
28  ;TO THE GPIB BUS.
29
30  ;INTERRUPT SERVICE ROUTINE--
31  ;CHECKS FOR INPUT OR OUTPUT INTERRUPTS AND HANDLES WHICHEVER
32  ;AS APPROPRIATE. IT ALSO CHECKS FOR ONE SPECIAL INPUT, THE
33  ;RESET (ABORT) FUNCTION, WHICH REINITIALIZES EVERYTHING.
34
35  ;GPIB REGISTERS REQUIRED:
36  ;GPVEC = 270 ;INTERRUPT VECTOR ADDRESS
37  ;GPISR = 164000 ;INTERRUPT SERVICE REGISTER (R0)
38  ;GPIMR = 164000 ;INTERRUPT MASK REGISTER (WO)
39  ;GPCTSR = 164001 ;CONTROL STATUS REGISTER (R0)
40  ;GPCSR = 164002 ;COMMAND STATUS REGISTER (R0)
41  ;GPANR = 164004 ;ADDRESS MODE REGISTER (WO)
42  ;GPACR = 164006 ;AUXILIARY CONTROL REGISTER (RW)
43  ;GPASWR = 164010 ;ADDRESS SWITCH REGISTER (R0)
44  ;GPADR = 164010 ;ADDRESS REGISTER (WO)
45  ;GPSPR = 164012 ;SERIAL POLL REGISTER (RW)
46  ;GPPCR = 164015 ;CONTROLLER COMMAND REGISTER (WO)
47  ;GPDIR = 164016 ;DATA IN REGISTER (R0)
48  ;GPDOR = 164016 ;DATA OUT REGISTER (WO)
49
50  ;PROM=1 ;SET TO 1 WHEN INTERRUPT VECTORS LOADED INTO PROM,
51  ;;OTHERWISE THEY ARE LOADED DYNAMICALLY BY GPINI.
52  ;RAM=0 ;ALSO, NO LOCAL RAM VARIABLES IF PROM=1.

```

```

54 000000 010046 : GPINI :: : **** GPIB INITIALIZATION SUBROUTINE *****
55 000000 010046 : MOV R0,-(SP) : SAVE REGISTER 0 ON STACK
56 000002 000005 : RESET : ?NEEDED?
57 000004 105037 164006 : CLR B : INITIALIZE GPIB AUX REGISTER
58 000004 : : : SET HOLD RD ON EO1
59 113700 164010 : NOVB : GET CONTROLLER ADDRESS
60 000010 032700 000149 : BIT : TEST CERTAIN SWITCHES ON CONTROLLER
61 000014 061401 : BEQ 58 : : OKAY! CONTINUE
62 000029 061401 : HALT : ERROR: SAC AND/OR EXT SET
63 000022 000000 : BIC R0,eCPADR : ENABLE TALKING AND LISTENING
64 000024 042700 177740 : NOVB : LOAD CONTROLLER ADDRESS
65 000030 116037 164010 : .IF EQ PRTN : DON'T DO IF VECTOR SPACE IS ROM
66 : : : NOV $1SHCP, eCPVEC : LOAD INTERRUPT VECTOR
67 : : : NOV $340, eCPVEC+2 : AND PSW
68 : : : END.C : RESTORE REGISTER 0
69 012600 : (SP)+,R0 : LOAD ADDRESS TO GO IF TRIGGER
70 000034 016537 : NOV 2(R5),eCTRIG : LOAD ADDRESS FROM LAST INPUT
71 000036 016537 : NOV 2(R5),ePEOF : CLEAR EO1 FLAG
72 000044 005037 000000C : CLR eCPNCH : NUMBER OF CHARACTERS RECEIVED
73 000059 112737 000240 164000 : NOVB eCPIMR : ENABLE GET INTERRUPTS
74 000056 112737 000010 164015 : NOVB eCPCCR : ENABLE MASTER INTERRUPT
75 000064 000297 : RTS PC : BUS IS NOW INITIALIZED
76 : : : **** GPIB STRING INPUT SUBROUTINE *****
77 000066 : GPINI :: : **** GPIB STRING INPUT SUBROUTINE *****
78 000066 : BISB : RELEASE RD HANDSHAKE FROM LAST INPUT
79 016537 000002 000000C : MOV 2(R5),eGPACR : INPUT BUFFER ADDRESS
80 000066 016537 000004 000000C : NOV eGPBIN : MAXIMUM LENGTH ALLOWED
81 000074 017537 : CLR eCPMAX : NUMBER OF CHARACTERS RECEIVED
82 000102 005037 000000C : NOVB eCPNCH : ENABLE BI INTERRUPTS
83 000106 112737 000241 164000 : .IF : WAIT eCPIMR : WAIT FOR 'ANY' INTERRUPT
84 000114 000001 : TST eCPEOF : END-OF-INPUT? YES
85 000116 005037 000000C : BNE 208 : eCPMAX,eCPNCH : BUFFER FULL?
86 000122 001004 : CMP BCT 108 : NOT DONE/NOT FULL
87 000124 023737 000000C 000000C : NOV eCPIMR : DISABLE BI INTERRUPTS
88 000132 003370 : CLR eCPEOF : RETURN NUMBER OF CHARACTERS
89 : : : NOV eCPNCH, e6(R5) : INPUT COMPLETE
90 000134 112737 000240 164000 : 208 : **** GPIB STRING OUTPUT SUBROUTINE *****
91 000142 005037 000000G 0000006 : NOV eCPNCH : RELEASE RD HANDSHAKE FROM LAST INPUT
92 000146 013775 000000C 0000006 : CLR eCPNCH : GET OUTPUT BUFFER ADDRESS
93 000154 000207 : RTS PC : GET NUMBER BYTES TO OUTPUT
94 : : : NOVB eCPNCH : ENABLE BO INTERRUPTS
95 : : : NOVB eCPNCH : SET SRQ AND READ BIT
96 000156 : BIGB : ALL CHARACTERS OUTPUT?
97 016537 000002 000000C : NOV eCPNCH : NO--WAIT MORE
98 017537 000004 000000C : NOVB eCPNCH : DISABLE BO INTERRUPTS
99 000156 016537 000002 000000C : NOV eCPNCH : ** HOST DOESN'T CLEAR '2'
100 000164 017537 000004 000000C : NOVB eCPNCH : OUTPUT COMPLETE
101 000172 112737 000349 164000 : NOVB eCPNCH : **** GPIB STRING OUTPUT SUBROUTINE *****
102 000209 112737 000102 164012 : NOVB eCPNCH : **** GPIB STRING OUTPUT SUBROUTINE *****
103 000206 000001 : NOVB eCPNCH : **** GPIB STRING OUTPUT SUBROUTINE *****
104 000210 005037 000000C : NOVB eCPNCH : **** GPIB STRING OUTPUT SUBROUTINE *****
105 000214 003374 000240 164000 : NOVB eCPNCH : **** GPIB STRING OUTPUT SUBROUTINE *****
106 000216 112737 000240 164000 : NOVB eCPNCH : **** GPIB STRING OUTPUT SUBROUTINE *****
107 000224 105037 164012 : NOVB eCPNCH : **** GPIB STRING OUTPUT SUBROUTINE *****
108 000236 000207 : RTS PC : **** GPIB STRING OUTPUT SUBROUTINE *****

```

```

116      000232    032737    001000    164000  ISRGP:: :**** GPIB I/O INTERRUPT SERVICE ROUTINE *****
117      000232    001407    000002    164015  108:   BIT    *1000,@@GPISR    ;TEST IFC BIT
118      000240    001407    000002    164015  BEQ    208    @@GPCCR    ;OKAY IF NOT SET
119      000242    142737    000002    164015  BICB   *2,@@GPCCR    ;CLEAR IFC BIT
120      000256    132737    000002    164015  BITB   *2,@@GPCCR    ;SUCCESSFUL?
121      000256    001371    BNE    108    ;NO--TRY AGAIN
122      000300    132737    000001    164000  208:   BITB   *1,@@GPISR    ;"BI" INTERRUPT SET?
123      000300    132737    000001    164000  BNE    408    ;YES
124      000310    606436    BNE    408    ;NO--CONTINUE
125      000312    000177    000000C   308:   JMP    @@TRIG    ;RESTART
126      000316    132737    000002    164000  408:   JMP    @@TRIG    ;IF DATA RECEIVED FROM BUS
127      000316    001403    000001    000000C  BEQ    *2,@@GPISR    ;END-OF-INPUT?
128      000324    001403    000001    000000C  418:   MOV    418    ;NO--CONTINUE
129      000326    012737    000001    000000C  MOVB   *1,@@GPEOF    ;SET EOI FLAG
130      000334    113777    164016    000000C  MOVB   @@GPDIR,@@GPBIN    ;GET A CHARACTER
131      000342    005237    000000C   418:   INC    @@GPBIN    ;INCREMENT INPUT ADDRESS
132      000346    005237    000000C   418:   INC    @@GPNCH    ;INCREMENT CHARACTER COUNT
133      000352    000415    BNE    1002    ;NO--NEXT?
134      000352    000415    BR    1002    ;DATA-OUT REGISTER CLEAR?
135      000354    117737    000000C   164016  508:   MOVB   @@GPBOUT,@@GPDOR    ;IF DATA TO SEND TO BUS
136      000354    005237    000000C   164016  INC    @@GPBOUT    ;SEND OUT DATA BYTE
137      000362    005237    000000C   164016  DEC    @@GPNCH    ;INCREMENT OUTPUT ADDRESS
138      000366    005337    000000C   164016  BLE    608    ;DONE WITH STRING?
139      000372    003491    000000C   164016  BR    1008    ;YES--CLEANUP
140      000374    006464    000000C   164016  BITB   *100,@@GPISR    ;NO--NEXT?
141      000404    001774    0000002   1008:  BEQ    608    ;DATA-OUT REGISTER CLEAR?
142      000404    001774    RTI    ;NO
143      000406    0000002   1008:  RTI    ;NO
144      000406    0000002   1008:  RTI    ;ASSEMBLE IF VARIABLES NOT PROM
145      000406    0000002   1008:  RTI    ;E-O-I FOUND? FLAG
146      000406    0000002   1008:  RTI    ;INPUT BUFFER ADDRESS
147      000406    0000002   1008:  RTI    ;OUTPUT BUFFER ADDRESS
148      000406    0000002   1008:  RTI    ;MAXIMUM CHARACTERS ON INPUT
149      000406    0000002   1008:  RTI    ;CHARACTER COUNT FOR INPUT OR OUTPUT
150      000406    0000002   1008:  RTI    ;ADDRESS TO GO IF "GET" RECEIVED
151      000406    0000002   1008:  RTI    ;ENDC
152      000406    0000002   1008:  RTI    ;END
153      000406    0000002   1008:  RTI    ;END
154      000406    0000002   1008:  RTI    ;END
155      000406    0000002   1008:  RTI    ;END
156      000406    0000002   1008:  RTI    ;END

```

(7) MODULE GRMS - This subroutine computes parameters from statistical techniques.

Calling Sequence:

CALL GRMS (N, X, RMS, DC, PEAKMX, PEAKMN)

where the arguments are:

N	Length of sequence
X	Floating-point data sequence
RMS	Returned rms voltage (standard deviation)
DC	Returned dc voltage (average)
PEAKMX	Peak voltage (maximum)
PEAKMN	Peak voltage (minimum)

NOTE: This code was originally written by R.W. Anderson and subsequently modified by R.W. Luckey, both of NRL-USRD.

```

1   2
2   3   4   5   6   7   8   9   10
3   4   5   6   7   8   9   10
4   5   6   7   8   9   10
5   6   7   8   9   10
6   7   8   9   10
7   8   9   10
8   9   10
9   10
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
      TITLE GRMS
      IDENT /62/
      :SUBROUTINE TO COMPUTE TIME-AVERAGE VOLTAGES: RMS, DC,
      :MAXIMUM, AND MINIMUM.
      :
      :AUTHOR: RW ANDERSON (SOMETIME IN 1980)
      :MODIFIED: RW LUCKY (FOR BOTH PEAKS, 1981)
      :
      :FORTRAN CALL--:
      :CALL GRMS (NPTS, VALS, RMS, AVG, PEAKMX, PEAKMN)
      :
      :AC0 - IS THE SUMMING REGISTER
      :AC1 - IS THE REGISTER FOR THE SUM OF THE SQUARES
      :AC2 -
      :AC3 - NUMBER OF POINTS
      :AC4 - PEAK MINIMUM - STORAGE - REGISTER NOT GENERALLY USABLE
      :AC5 - PEAK MAXIMUM - STORAGE - REGISTER NOT GENERALLY USABLE
      :
      :R0 - LOOP COUNTER - INITIALIZED TO NPTS
      :R1 - ADDRESS OF THE VALUES
      :
      :R00000 1700001 0000004
      :R00002 172475 0000004
      :R00006 174004 0000004
      :R00010 172475 0000004
      :R00014 174005 0000004
      :R00016 017599 0000002
      :R00022 177300 0000002
      :R00024 016591 0000004
      :R00030 176400 0000004
      :R00032 176401 0000004
      :R00034 172611 0000002
      :R00036 172002 0000001
      :R00040 171202 0000001
      :R00042 172102 0000002
      :R00044 172621 0000004
      :R00046 173605 0000005
      :R00050 178900 0000000
      :R00052 100001 0000001
      :R00054 174205 0000005
      :R00056 173604 0000004
      :R00060 176000 0000000
      :R00062 160401 0000001
      :R00064 174204 0000004
      :R00066 077916 0000000
      :
      CRMS:: SETF   04(R5), AC0
      STF    AC0, AC4
      LDF    04(R5), AC0
      STF    AC0, AC5
      MOV    02(R5), R0
      LDCLIF R0, AC3
      MOV    4(R5), R1
      CLRF   AC0
      CLRIF AC1
      LDF    (R1), AC2
      ADDF   AC2, AC9
      HULF   AC2, AC2
      ADDF   AC2, AC1
      LDF    (R1)+, AC2
      CMPF   AC5, AC2
      CFCC
      BPL   MIN, AC5
      STF   AC2, AC5
      CMPF   AC4, AC2
      CFCC
      BH1   CONT
      STF   AC2, AC4
      SOB   R0, LOOP
      :
      :PEAK MINIMUM = X(1)
      :STORE IN AC4
      :PEAK MAXIMUM = X(1)
      :STORE IT IN AC5
      :SET UP LOOP COUNTER
      :NUMBER OF POINTS (NPTS)
      :ADDRESS OF DATA IN REGISTER ONE
      :ZERO SUMMING REGISTER
      :ZERO SQUARES SUMMING REGISTER
      :GET A VALUE
      :ADD IT TO SUMMING REGISTER
      :SQUARE IT
      :ADD IT TO SQUARES SUMMING REGISTER
      :GET THE VALUE AGAIN
      :SEE IF NEW MAX PEAK
      :COPY FUNCTION CODES
      :BRANCH IF NOT NEW MAX PEAK
      :SAVE NEW MAX PEAK
      :SEE IF NEW MIN PEAK
      :COPY FUNCTION CODES
      :BRANCH IF NOT NEW MIN PEAK
      :SAVE NEW MIN PEAK
      :LOOP IF NOT FINISHED
      :
      MIN:
      :
```

CENSUS MACRO M1113 23-JUL-82 14:15 PAGE 2

55	6000970	174403
56	6000972	174075
57	6000976	172605
58	6001000	173200
59	6001022	174270
60	600106	172604
61	600110	173200
62	600112	174270
63	600116	171900
64	600120	174503
65	600122	173100
66	600124	174100
67	600126	094767
68	600132	174075
69	600136	600297
		600001
		79

```

DIVF AC3,          ! AVERAGE (AVG) = SUM / NPTS
      AC9,          ! STORE AVERAGE (AVG) - DC
      STF,          ! MOVE MAX PEAK INTO AC2
      AC2,          ! PEAK MAX = PEAK MAX - AVG (DC)
      AC2,          ! STORE PEAK MAX
      SUBF AC2,      ! MOVE MIN PEAK INTO AC2
      AC2,          ! PEAK MIN = PEAK MIN - AVG (DC)
      STF,          ! STORE PEAK MIN
      AC9,          ! SQUARE THE AVERAGE (DC)
      AC1,          ! SQUARE AVERAGE = SUMSQ / NPTS
      AC1,          ! SUBTRACT AVERAGE SQUARED (DC)**2
      AC9,          ! PUT IT IN AC9
      JSR PC,        ! GET THE SQUARE ROOT
      STF AC9,       ! STORE RMS
      RTS PC,        ! RETURN FROM SUBROUTINE
      END

```

(8) LSIVAR - The following is an alphabetical list (with definitions) of the variables used in the software. All are used by LSIVM except for those used by the subroutines noted in parentheses.

AO	Exponent factor (DGZL)
AD1C	Input port for channel 1
AD2C	Input port for channel 2
AD3C	Input port for channel 3
ADBUF	A/D buffers (three contiguous arrays)
ADS	Numbers of A/D's used
ADST	Measurement status register (A32D)
ADT	DMA to interrupt on (ignore all others)
ADUM	A/D pseudo-register (A32D)
ADWL	Length in bytes of buffer to send to host
AERRS	Automatic retries when A/D errors occur
AVGCOD	Averaging mode (0 = don't, or cycles per averaged subsequence)
BSIZE	Buffer size allowed per channel
BUF1	12-bit integer values from A/D #1
BUF2	12-bit integer values from A/D #2
BUF3	12-bit integer values from A/D #3
C0	Cosine factor (DGZL)
C1	Cosine factor (DGZL)
CFACT	Current factor ( $I = CFACT \cdot E$ )
CHERRY	Display character buffer
CHNO	Channel number being computed
COMTYP	Computation type (not used)
COV	Covariance (ac power)
DATA	Start of LSIVM input parameter buffer

DCV	Average (dc) voltage from statistical computation
DFD	DFT data array
DIST1	Distortion of channel 1
DIST2	Distortion of channel 2
DIST3	Distortion of channel 3
DIST	Harmonic distortion
DLUN	Display device (1 = terminal, 2 = display)
DSTYLE	Display style (0-5)
ECNT	Counter for A/D error retry
FSAM	Sample frequency (Hz)
FSIG	Signal frequency (Hz)
GAIN1	A/D gain channel 1 (1, 2, 5, 10)
GAIN2	A/D gain channel 2 (1, 2, 5, 10)
GAIN3	A/D gain channel 3 (1, 2, 5, 10)
GAIN	A/D converter gains array
GATED	Gated voltage flag (0 = no, 1 = yes)
GPBIN	Input buffer address (GPIOSR)
GPBOUT	Output buffer address (GPIOSR)
GPEOF	E-O-I found - flag (0 = no, 1 = yes) (GPIOSR)
GPMAX	Maximum characters on input (GPIOSR)
GPNCH	Character count for input or output (GPIOSR)
GTRIG	Address to go to if "GET" received (GPIOSR)
HAR	Number of harmonics to use to compute distortion
HARM	Harmonics - used flag array (DAPD)
HTC	Specific harmonic (line + 1) to compute
I	Line number (DAPD)
IBLEN	Bytes received from IEEE-488 bus

IC	Command byte 1
ICA	Count of A/D conversions
ICE	Count of A/D errors
ICN	Command byte 2 (may be channel number)
IERTD	Error status of DFT
INBUF	IEEE-488 bus input packet
IP	Command parameter string
ISW	LSIVM status word
IWD	What DMA to interrupt from (A32D)
LADD	Pointer to displayed parameters
LCTR	Counter for displayed parameters
LDAT	Parameter to display
LPTR	Index for displayed parameters
NCAV	Sequences to average (NCYC/AVGCOD)
NCYC	Cycles per sequence
NCYF	Cycles to compute (AVGCOD); averaging changes
NCYHA	Cycles to compute (for a harmonic)
NSAM	Samples per sequence (DAPD)
PAD	Samples to convert per DMA transfer per channel
PDFT	Phase from DFT
PEKVM	Peak (-) voltage from inspection
PEKVP	Peak (+) voltage from inspection
PHAS1	Phase of channel 1
PHAS2	Phase of channel 2
PHAS3	Phase of channel 3
POWER	Total power (ac and dc)
PSIGN	Phase sign (DAPD)

PTC	Points to compute (PTR/NCYC); averaging changes this
PTR	Samples to transfer to host (samples per sequence)
PWR	Harmonic power (DAPD)
RAA	Argument list for writing buffer to host
RAM	Start of program RAM space
RAN	Address of buffer to send to host
RBUF	One voltage buffer (converted to floating-point)
RMD	Statistical data array
RMSV1	RMS voltage in channel 1
RMSV2	RMS voltage in channel 2
RMSV3	RMS voltage in channel 3
RMSV	RMS voltage from statistical analysis
RPHAS	Phase difference between channels 1 and 2
SO	Sine factor (DGZL)
SJM	Second joint moment (total power)
STACK	Start (top) of program stack
STPT	Number of first sample to transfer to host
STRING	Used by print
T1	DMA #1 pseudo-register (A32D)
T2	DMA #2 pseudo-register (A32D)
T3	DMA #3 pseudo-register (A32D)
VDFT	Total rms voltage from DFT
WAD	What A/D converters used (A32D)
WAV	Average window magnitude
WIND	Window buffer (floating-point)
WMODE	Window mode (0 = no, 1 = yes)
XIMAG	Imaginary voltage part (DAPD)

XREAL

Real voltage part (DAP<sub>u</sub>,

```

1      .TITLE LSIVAR
2      .IDENT /06/
3      .GLOBAL VARIABLES (DYNAMIC) FOR LSIVM
4
5
6      .PSECT $$VARL.REL.RO !NOT REALLY RO!! TO FOOL TASKBUILDER!!
7      .BLKB 2452 !SHIFT BEGINNING OF RAM TO 4K BOUNDARY
8
9
10     00230000
11     002452    000000
12     002452    000000
13     003454    000000
14     003454    000000
15     003456    000000
16     003456    000000
17
18     003460    000
19     003460    000
20     003461    000
21     003462    000000
22     003602    000000
23     003604    000000
24     003606    000000
25
26     003610    000000
27     003612    000000
28     003614    000000
29     003616    000000
30     003620    000000
31     003622    000000
32
33     003624    000000
34     003624    000000
35     003626    000000
36     003626    000000
37     003630    000000
38     003632    000000
39     003634    000000
40     003636    000000
41     003640    000000
42     003642    000000
43     003644    000000
44     003646    000000
45     003650    000000
46     003652    000000
47     003654    000000
48     003656    000000
49     003660    000000
50     003662    000000
51     003664    000000
52     003666    000000
53     003670    000000
54     003672    000000
55     003674    000000
      .ISIZE=1024.          !BUFFER SIZE ALLOWED PER CHANNEL
      RAM:::WORD 0           !START OF PROGRAM RAM SPACE
      STACK:::WORD 0         !START (TOP) OF PROGRAM STACK
      ICA:::WORD 0           !COUNT OF A/D CONVERSIONS
      ICE:::WORD 0           !COUNT OF A/D ERRORS
      INBUF:::BYTE 0          !IEEE-488 BUS INPUT PACKET
      IC:::BYTE 0             !COMMAND BYTE 1
      ICM:::BYTE 0            !COMMAND BYTE 2 (MAYBE CHANNEL NUMBER)
      IP:::WORD 40.           !COMMAND PARAMETER STRING
      ISW:::WORD 0             !LSIVM STATUS WORD
      IBLEN:::WORD 0           !BYTES RECEIVED FROM IEEE-488 BUS
      CHNO:::WORD 0           !CHANNEL NUMBER BEING COMPUTED
      GEOF:::WORD 0            !E-O-I FOUND? FLAG (0=NO, 1=YES) (GP10SR)
      GPBIN:::WORD 0           !INPUT BUFFER ADDRESS (GP10SR)
      GPBOUT:::WORD 0          !OUTPUT BUFFER ADDRESS (GP10SR)
      GPMAX:::WORD 0           !MAXIMUM CHARACTERS ON INPUT (GP10SR)
      GPNCH:::WORD 0           !CHARACTER COUNT FOR INPUT OR OUTPUT (GP10SR)
      GTRIG:::WORD 0           !ADDRESS TO GO TO IF "GET" RECEIVED (GP10SR)
      DATA:::WORD 0             !START OF LSIVM INPUT PARAMETER BUFFER
      GAIN:::WORD 0             !START OF A/D CONVERTER GAINS ARRAY
      GAIN1:::WORD 0            !A/D GAIN CHANNEL 1 (1..2..5..10)
      GAIN2:::WORD 0            !A/D GAIN CHANNEL 2 (1..2..5..10)
      GAIN3:::WORD 0            !A/D GAIN CHANNEL 3 (1..2..5..10)
      PTR:::WORD 0              !SAMPLES TO TRANSFER TO HOST (SAMPLES PER SEQUENCE)
      STPT:::WORD 0             !NUMBER OF FIRST SAMPLE TO TRANSFER TO HOST
      ADS:::WORD 0              !NUMBERS OF A/D'S USED (1=1..2..2..4..3..7=ALL.)
      ADT:::WORD 0              !DMA TO INTERRUPT ON (IGNORE ALL OTHERS)
      PAD:::WORD 0              !SAMPLES TO CONVERT PER DMA TRANSFER PER CHANNEL
      NCYC:::WORD 0             !CYCLES PER SEQUENCE
      HAR:::WORD 0              !NUMBER OF HARMONICS TO USE TO COMPUTE DISTORTION
      AVGOD:::WORD 0            !AVERAGING MODE (0=DON'T, OR CYCLES PER AVERAGE SUBSEQ..)
      DSTYLE:::WORD 0           !DISPLAY STYLE (0..5)
      COMTYP:::WORD 0           !COMPUTATION TYPE (NOT USED)
      AERRS:::WORD 0             !AUTOMATIC RETRIES WHEN A/D ERRORS OCCUR
      GATE:::WORD 0              !GATED VOLTAGE FLAG (0=NO, 1=YES)
      WMODE:::WORD 0             !WINDOW MODE (0=NO, 1=YES)
      DLUN:::WORD 0              !DISPLAY DEVICE (1=TERMINAL, 2=DISPLAY)
      HTC:::WORD 0               !SPECIFIC HARMONIC (LINE+1) TO COMPUTE
      AD1C:::WORD 0              !INPUT PORT FOR CHANNEL 1
      AD2C:::WORD 0              !INPUT PORT FOR CHANNEL 2
      AD3C:::WORD 0              !INPUT PORT FOR CHANNEL 3

```

LISIIVAR MACNO M113 20-SEP-02 09:56 PAGE 2

```

57 003676 000000 000000 000000 ! SIGNAL FREQUENCY (HZ)
58 003702 000000 000000 000000 ! SAMPLE FREQUENCY (HZ)
59                                     ! POINTS TO COMPUTE (PTR/MCYC), AVERAGING CHANGES
60 003706 000000 000000 000000 ! CYCLES TO COMPUTE (AVGCD), AVERAGING CHANGES
61 003710 000000 000000 000000 ! CYCLES TO COMPUTE (FOR A HARMONIC)
62 003712 000000 000000 000000 ! SEQUENCES TO AVERAGE (MCYC/AVGCD)
63 003714 000000 000000 000000 ! COUNTER FOR A/D ERROR RETRY
64 003716 000000 000000 000000 ! LENGTH IN BYTES OF BUFFER TO SEND TO HOST
65 003720 000000 000000 000000

66                                     ! START OF DFT DATA ARRAY
67 003722 000000 000000 000000 ! TOTAL RMS VOLTAGE FROM IFFT
68 003722 000000 000000 000000 ! PHASE FROM DFT
69 003726 000000 000000 000000 ! HARMONIC DISTORTION

70                                     ! START OF TIME-INTEGRATION DATA ARRAY
71 003736 000000 000000 000000 ! RMS VOLTAGE FROM TIME INTEGRATION
72 003736 000000 000000 000000 ! PEAK(+) VOLTAGE FROM INSPECTION
73 003742 000000 000000 000000 ! PEAK(-) VOLTAGE FROM INSPECTION
74 003742 000000 000000 000000 ! AVERAGE (DC) VOLTAGE FROM TIME INTEGRATION
75 003746 000000 000000 000000
76 003752 000000 000000 000000
77                                     ! TOTAL POWER (SUM OF E*I)
78 003756 000000 000000 000000 ! CURRENT FACTOR (I=CFACT*E)
79 003762 000000 000000 000000
80                                     ! SECOND JOINT MOMENT
81 003766 002000 000000 000000 ! COVARIANCE
82 003772 002000 000000 000000
83                                     ! ARGUMENT LIST FOR WRITING BUFFER TO HOST
84 003776 000030 000000 000000 ! ADDRESS OF BUFFER TO SEND TO HOST
85 004004 000000 000000 000000 ! RAM VARIABLES USED BY A32D
86                                     ! WHAT A/D CONVERTERS USED (A32D)
87 004006 000000 000000 000000 ! WHAT DMA TO INTERRUPT FROM (A32D)
88 004010 000000 000000 000000 ! DMA #1 PSEUDO-REGISTER (A32D)
89 004012 000000 000000 000000 ! DMA #2 PSEUDO-REGISTER (A32D)
90 004014 000000 000000 000000 ! DMA #3 PSEUDO-REGISTER (A32D)
91 004016 000000 000000 000000 ! I/A/D PSEUDO REGISTER (A32D)
92 004020 000000 000000 000000 ! MEASUREMENT STATUS REGISTER (A32D)
93 004022 000000 000000 000000 ! CHANNEL 1 PORT ADDRESS * 400
94 004024 000000 000000 000000 ! CHANNEL 2 PORT ADDRESS * 400
95 004026 000000 000000 000000 ! CHANNEL 3 PORT ADDRESS * 400
96 004030 000000 000000 000000
97                                     ! COUNTER FOR DISPLAYED PARAMETERS
98 004032 000000 000000 000000 ! INDEX FOR DISPLAYED PARAMETERS
99 004034 000000 000000 000000 ! POINTER TO DISPLAYED PARAMETERS
100 004036 000000 000000 000000 ! PARAMETER TO DISPLAY PARAMETERS
101 004049 000000 000000 000000 ! STRING : .BLKW 6
102 004042 000000 000000 000000

```

```

105      004056    000000    000000    000000    XREAL: : .FLT2   0.0
106      004062    000000    000000    000000    XIMAG: : .FLT2   0.0
107      004062    000000    000000    000000    PWR: : .FLT2   0.0
108      004066    000000    000000    000000    PSIGN: : WORD    0
109      004072    000000    000000    000000    1: : WORD    0
110      004074    000000    000000    000000    NSAM: : WORD    0
111      004076    000000    000000    000000    HARM: : .BLKW   20.
112      004100    000000    000000    000000    1END: : .WORD    0
113      004150    000000    000000    000000    1END: : .WORD    0
114      004150    000000    000000    000000    ! ERROR STATUS OF DFT

```

LSIVAR MACRO M1113 20-SEP-82 09:56 PAGE 4

```

116      004152    000000    000000    000000    XREAL: : .FLT2   0.0
117      004160    000000    000000    000000    XIMAG: : .FLT2   0.0
118      004162    000000    000000    000000    PWR: : .FLT2   0.0
119      004176    000000    000000    000000    PSIGN: : WORD    0
120      004172    000000    000000    000000    1: : WORD    0
121      004200    000000    000000    000000    NSAM: : WORD    0
122      004202    000000    000000    000000    HARM: : .BLKW   20.
123      004210    000000    000000    000000    1END: : .WORD    0
124      004256    000000    000000    000000    CHERY: : .BLKB   36.
125      004262    000000    000000    000000    RMSV1: : .FLT2   0.0
126      004266    000000    000000    000000    RMSV2: : .FLT2   0.0
127      004272    000000    000000    000000    RMSV3: : .FLT2   0.0
128      004276    000000    000000    000000    PHASI: : .FLT2   0.0
129      004302    000000    000000    000000    PHAS2: : .FLT2   0.0
130      004306    000000    000000    000000    PHAS3: : .FLT2   0.0
131      004312    000000    000000    000000    DIST1: : .FLT2   0.0
132      004316    000000    000000    000000    DIST2: : .FLT2   0.0
133      004322    000000    000000    000000    DIST3: : .FLT2   0.0
134      004326    000000    000000    000000    RPHAS: : .FLT2   0.0
135      004326    000000    000000    000000    ADBUF: : .BLKW   BSIZE
136      010326    000000    000000    000000    BUF1: : .BLKW   BSIZE
137      014326    000000    000000    000000    BUF2: : .BLKW   BSIZE
138      014326    000000    000000    000000    BUF3: : .BLKW   BSIZE
139      020326    000000    000000    000000    RBUF: : .BLKW   2*BSIZE
140      030326    000000    000000    000000    WAV: : .FLT2   0.0
141      030326    000000    000000    000000    WIND: : .BLKW   2*BSIZE
142
143      000001    000000    000000    000000    .END

```

(BLANK PAGE)

**APPENDIX F**  
**Voltmeter Memory Map**



Fig. F1 - Voltmeter memory map.

The ROM code used by this version of the voltmeter occupies approximately 15300 (octal) bytes of memory, so there is room for expansion.

The program variable space allows, at present, data buffers of 1024 samples, but this can be expanded by modifying the parameter BSIZE in LSIVAR when the program is rebuilt. Note that for each sample a buffer is expanded, memory is used as follows:

6 bytes for the sampled A/D waveform (2 bytes per channel),  
4 bytes for the integer to floating-point conversion buffer,  
4 bytes for the floating-point data window.

This requires approximately 7K words of RAM memory for each additional 1024 samples.